МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере
(направленность (профиль)/специализация)

БАКАЛАВРСКАЯ РАБОТА

на тему «Разработка автоматизированной системы учета пациентов стационара больницы»

Студент Е.А. Фролова _____
(И.О. Фамилия) (личная подпись)

Руководитель д.т.н., профессор, А.И. Туищев _____
(И.О. Фамилия) (личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)

«_____» _____ 20___ г.

Тольятти 2019

Аннотация

Тема данной выпускной квалификационной работы: «Разработка автоматизированной системы учета пациентов стационара больницы».

Целью выпускной квалификационной работы является разработка автоматизированной системы учёта пациентов стационара больницы.

Объект исследования – процессы хранения, обработки и ограничения доступа к электронным версиям документов, а так же создание сервисов по резервированию, восстановлению и учету доступа пользователей к документам.

Предмет исследования – автоматизация процесса работы с пациентами и автоматизация статистики хода обследования и лечения пациентов.

Пояснительная записка к выпускной квалификационной работе состоит из введения, трех глав, заключения, списка литературы и приложения.

Введение к выпускной квалификационной работе включает в себя описание цели и актуальности разработки системы, а так же краткую структуру выпускной квалификационной работы.

В пояснительной записке к выпускной квалификационной работе проанализированы процессы нахождения пациентов в стационаре с момента поступления до выписки, изучены принципы существующей организации процесса, выделены задачи, требующие автоматизации, рассмотрены варианты решения данных задач существующими программными средствами, выполнена постановка задания на разработку собственной системы автоматизации учёта пациентов стационара больницы.

В заключении представлены развёрнутые выводы по проделанной работе и её результатам.

Выпускная квалификационная работа включает: пояснительную записку – 47 страниц, 20 рисунков, 2 таблицы, и приложение.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Современные медицинские организации производят и накапливают огромные объемы данных. От того, насколько эффективно эта информация используется врачами, руководителями, управляющими органами, зависит качество медицинской помощи, общий уровень жизни населения, уровень развития страны в целом и каждого ее территориального субъекта в частности.

Целью работы является построение системы автоматизации бизнес-процессов ООО «Урарту» в целом и их стационара в частности.

К клиникам, основным видом деятельности которых является не только амбулаторное, но и стационарное лечение пациентов, относится и ООО «Урарту», где в ходе работы над ВКР была разработана и внедрена автоматизированной системы учёта пациентов стационара больницы.

Процесс автоматизации деятельности стационара включает такие этапы как разработка технического задания, описание и проектирование бизнес-процессов, создание базы данных, кодирование пользовательского интерфейса, тестирование и отладка системы автоматизации. Дальнейшее развитие системы автоматизации предусматривает анализ и сопровождение системы. На этапе эксплуатации системы также может потребоваться произвести перепроектирование системы автоматизации в соответствии с изменениями бизнес-структуры предприятия.

Использование клиент-серверных технологий является универсальным методом построения систем автоматизации малых и средних предприятий.

Цель работы - разработка автоматизированной информационной системы учета пациентов стационара, вследствие госпитализации.

О важности данной разработки говорит тот факт, что прикладное программное обеспечение информационных систем, необходимое учреждениям здравоохранения, редко может быть закуплено в готовом виде

и так как практически все учреждения здравоохранения обладают своей спецификой.

Выполнение выпускной квалификационной работы потребовало решения следующих основных задач:

- проведение анализа особенностей и специфики автоматизированных информационных систем в медицине;
- анализ технологии работы автоматизируемого подразделения;
- определение необходимой входной и выходной информации;
- разработка структуры базы данных;
- разработка и отладка программы;
- проведение опытной эксплуатации и внедрение у заказчика;
- определение экономического эффекта от внедрения разработанных программных средств.

Глава первая посвящена анализу предметной области и моделированию основных бизнес-процессов.

Глава вторая посвящена выбору технологии реализации и описанию реализации АИС.

Глава третья посвящена разработке, представлены экранные формы приложения.

ГЛАВА 1 Анализ предметной области

1.1 Описание предметной области

Администрация больницы заказала разработку автоматизированной системы учёта пациентов стационара больницы для отдела приема пациентов и медицинского секретариата. Новая система предназначена для обработки данных о врачах, пациентах, приеме пациентов и лечении. Система должна выдавать отчеты по запросу врачей или администрации. Во время предпроектного обследования составлено следующее описание деятельности рассматриваемых подразделений.

Перед приемом в стационар проводится встреча пациента и врача. Врач сообщает в отдел приема пациентов об ожидаемом приеме больного и передает данные о нем. Пациент может быть принят в больницу более чем один раз, но если пациент ранее не лечился в больнице, то ему присваивается регистрационный номер и записываются его данные (фамилия, имя и отчество, адрес и дата рождения). Пациент должен быть зарегистрирован в системе до приема в больницу.

Спустя некоторое время врач оформляет в «Отделе приема пациентов» прием больного. При этом определяется порядковый номер приема и заполняются данные приема пациента. После этого отдел приема посылает сообщение врачу для подтверждения приема больного. В это сообщение включается регистрационный номер пациента и его фамилия, порядковый номер приема, дата начала лечения и номер палаты.

В день приема пациент сообщает в приемное отделение о своем прибытии и передает данные о себе (или изменения в данных). Приемное отделение проверяет и при необходимости корректирует данные о пациенте. Если пациент не помнит свой регистрационный номер, то выполняется соответствующий запрос. После регистрации пациент получает регистрационную карту, содержащую ФИО пациента, адрес, дату рождения, номер телефона, группу крови, название страховой компании и номер страховки.

При необходимости врач запрашивает в медицинском секретариате историю болезни пациента, содержащую данные о курсах лечения, полученных пациентом.

Диаграмма DFD-модели предметной области автоматизированной системы учёта пациентов стационара больницы.

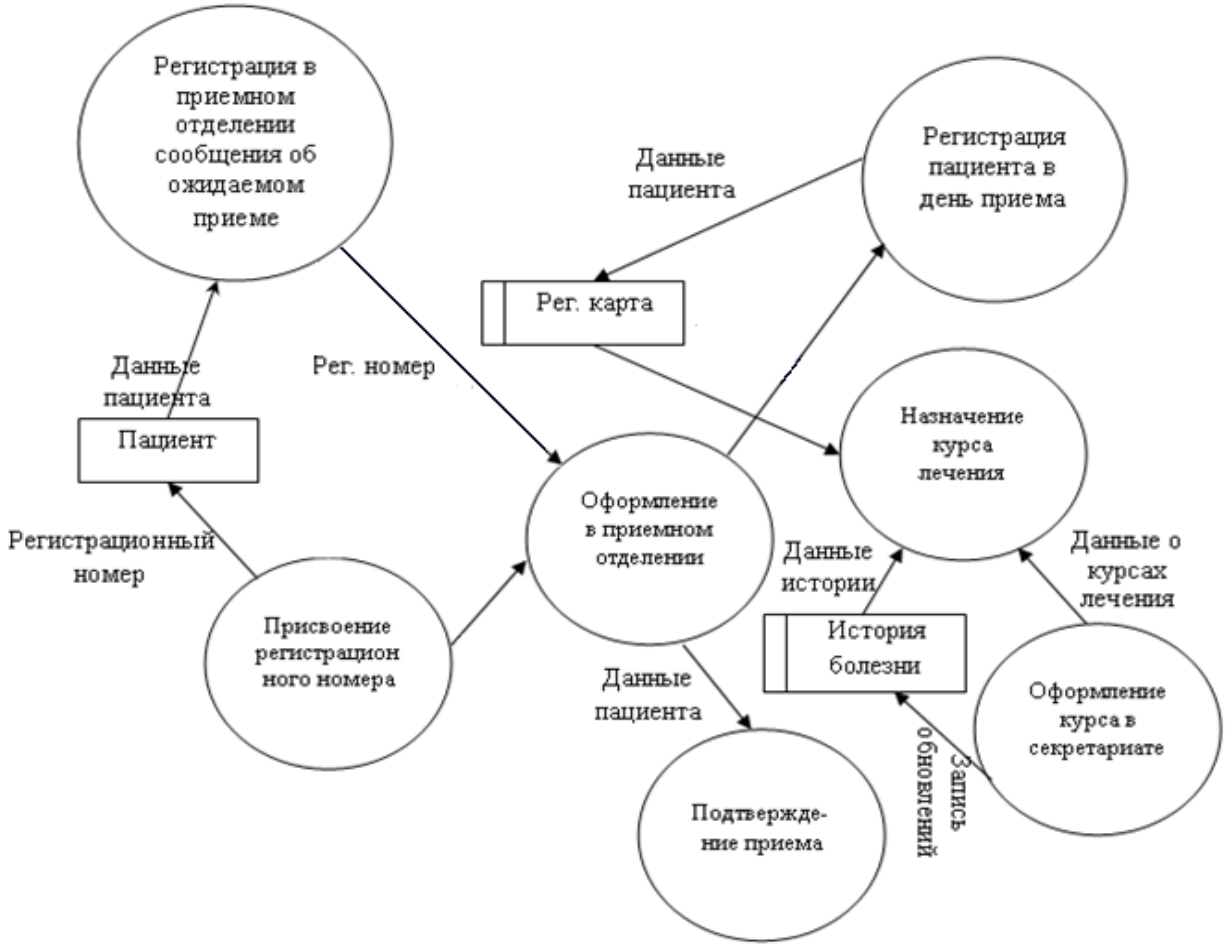

Рисунок 1 – «Диаграмма DFD модели»

Диаграмма потоков данных выявляет отношение между каждым процессом и показывает как пророцесс преобразует входные данные в выходные.

Диаграмма IDEF0-модели предметной области

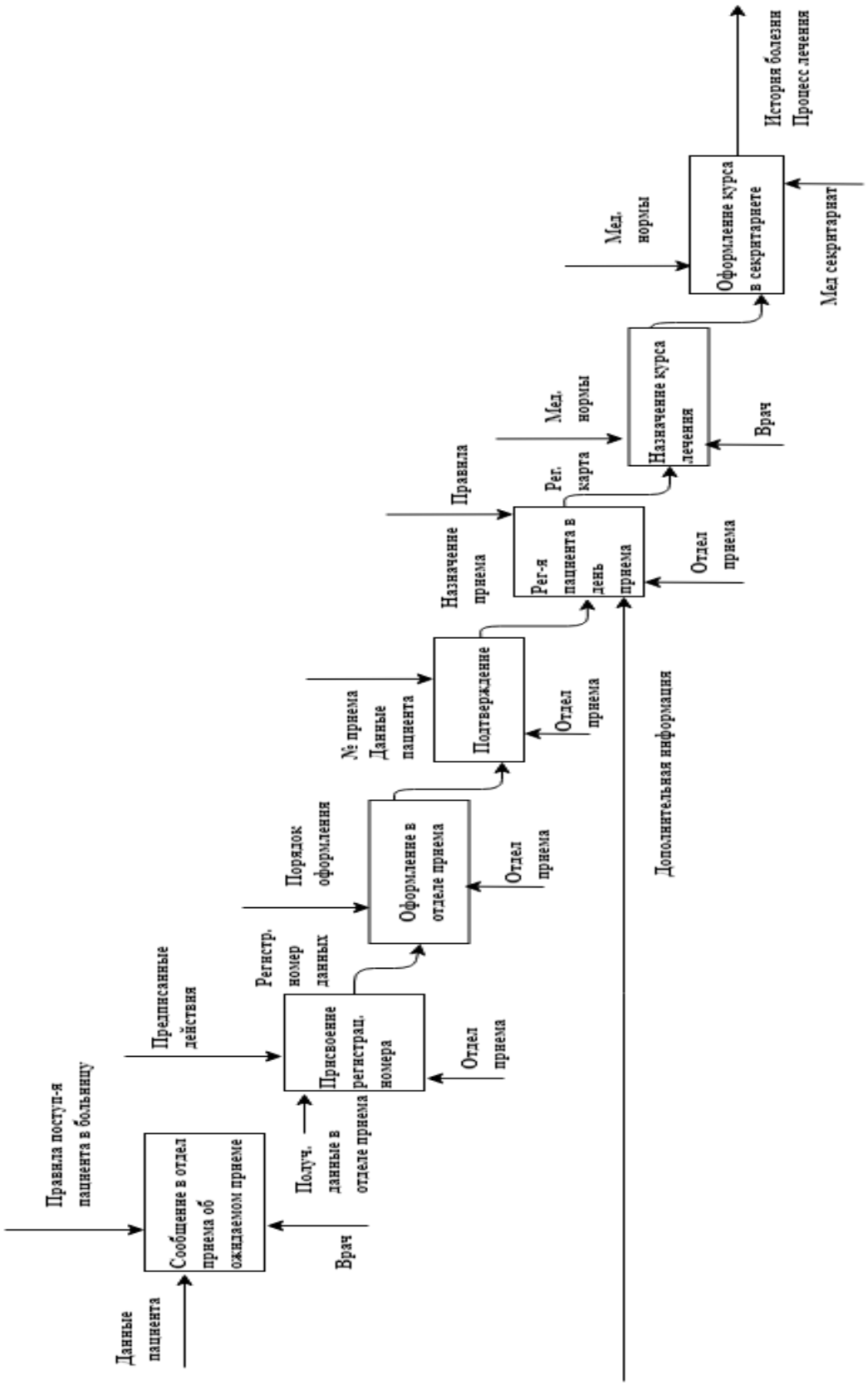

Рисунок 2 – «Диаграмма IDEF0 модели»

Диаграмма показывает состояние процесса до внедрения информационной системы и после.

1.2 Анализ существующих информационных систем в медицине

Система программного комплекса автоматизации амбулатории и стационара (ПК "АДИС") существует с 1992 года. В Самарской области система была внедрена с 19.06.1994. Впоследствии, шла постоянная модернизация данной системы.

ПК "АДИС" - это информационная технология, связанная с применением новейших средств вычислительной техники и связи, и использующая современные математические методы в задачах управления ресурсами.

ПК "АДИС" - это компетентный помощник медперсонала, от диспетчера до главного врача, достаточно простой в освоении и эксплуатации.

В 2018 году в эксплуатацию была внедрена новая «8.4» версия «ADIS», с обновленными справочниками, коррекцией профиля персонала, что существенно повлияло на оперативность работы и затруднило обработку статистических данных.

На ряду с системой АДИС используется автоматизированная система ведения истории болезней МИС "ЭИБ".

ПК «АДИС» можно представить в виде следующих взаимосвязанных подсистем:

- подсистемы оперативного режима – «Диспетчерская»;
- подсистемы ведения архива – «Архив»;
- подсистемы учета медикаментов – «Аптека»;
- подсистемы ведения справочников – «НСИ».

ПК «АДИС» поставляется в различных конфигурациях, учитывающих специфику организации малых, средних и больших городов.

Конкретное количество автоматизированных рабочих мест (АРМ) и их типов определяется потребностью городской станции скорой медицинской помощи.

Подсистема центральной диспетчерской реализуется как локальная сеть персональных компьютеров, а связь с удаленными подстанциями осуществляется через коммуникационный сервер на базе любой технологии передачи данных.

В отдельную категорию выделяются АРМ справочно-архивной службы и АРМ административного аппарата. Эти автоматизированные рабочие места предназначены для справочной и статистической обработки архивной информации (информации по обслуженным вызовам и хронологии работы бригад). В состав задач таких АРМ входит генератор отчетов и выборок, что позволяет получать практически произвольный набор отчетных форм на основе имеющейся в архиве информации и обеспечивает эффективный анализ работы станции за любой промежуток времени (от 1 часа до нескольких лет).

Рисунок 3 - Преимущества от использования автоматизированной системы ведения истории болезни

Медицинская информационная система "Электронная история болезни" позволяет:

– Заполнить в оперативном режиме паспортную часть истории болезни (в комплексе реализовано автоматическое заполнение личных данных пациента согласно сведениям, предварительно введенным в комплексе "АРМ администратора").

– Составить план обследования, включающий в себя: лабораторные исследования, функциональную диагностику, УЗИ и другие дополнительные обследования.

– Назначить консультации врачей-специалистов.

– Автоматически сформировать лист врачебных назначений на определенный срок согласно данным плана лечения, при необходимости - поменять сформированные назначения.

– Оперативно просматривать результаты лабораторных и инструментальных исследований (в графическом виде), консультации врачей-специалистов.

– Вести дневник лечащего врача.

– Формировать выписной эпикриз, в т.ч. в текстовый редактор MSWord или OpenOffice.org Writer.

– Использовать информацию из справочников, встроенных в комплекс (в том числе из МКБ -10), при вводе данных в электронную историю болезни, что значительно сокращает время на ее заполнение.

– Использовать заранее подготовленные шаблоны при вводе данных.

– Печатать любой из разделов электронной истории болезни.

– Формировать медицинские отчеты на основании данных истории болезни.

– Формировать самостоятельно любые запросы на основании данных истории болезни с передачей их в MSWord или MSExcel.

Преимущества использования автоматизированной системы ведения истории болезни представлены на рисунке 1.

Система АДИС используется в ООО Урарту с 2010 года. За это время возникла необходимость обновления программного обеспечения, добавление новых функций и задач. В связи с этим началась разработка новой информационно системы для учета пациентов стационара больницы.

Информационная взаимосвязь модулей МИС "ЭИБ" представлена на рисунке 2.

Предлагаемая информационная взаимосвязь модулей создает единое информационное пространство, которое, используя новейшие достижения в области информационных технологий и электронного документооборота, реализует максимальное снижение доли рутинной работы врачей, обеспечивает учет и высокое качество медицинских услуг, формирование всех необходимых форм отчетности в режиме реального времени.

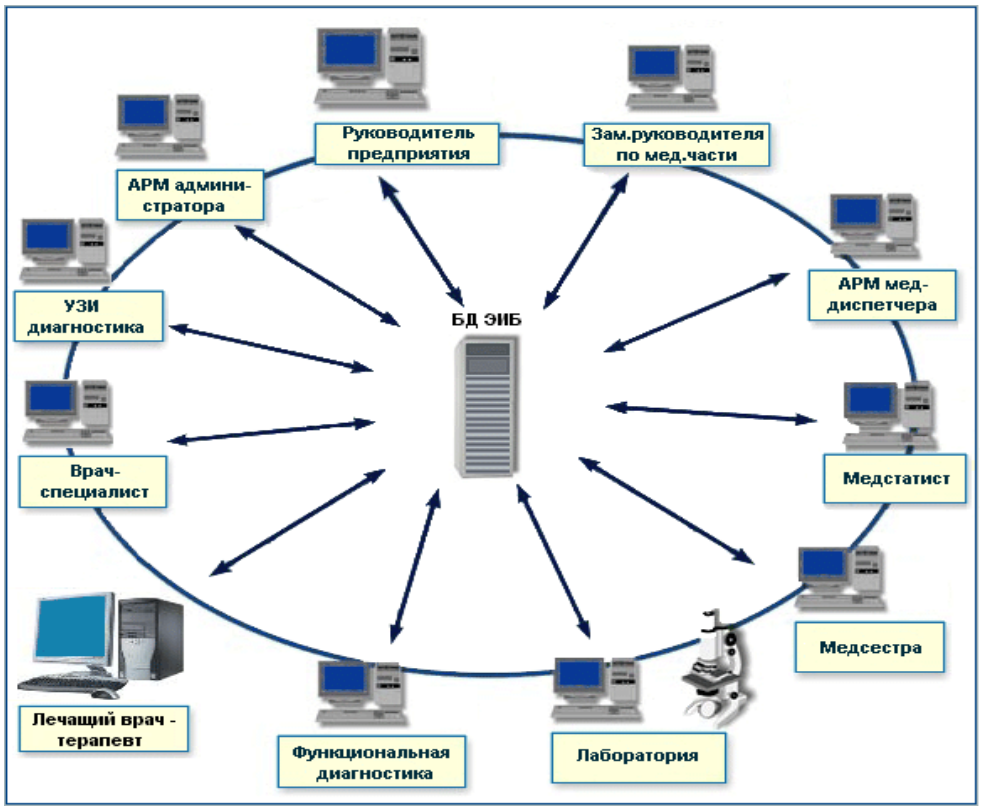

Рисунок 4 - Информационная взаимосвязь модулей МИС "ЭИБ"

1.3 Моделирование деятельности организации и информационных систем

Система предназначена для учета пациентов поступивших и выбывших со стационара, после их госпитализации.

Система не требует от пользователя дополнительных знаний по программированию, она предоставляет ему удобный экранный интерфейс для работы с многофайловой базой данных.

Рассмотрим работу системы. При госпитализации в систему вносятся данные о пациенте, такие как:

- Фамилия, имя, отчество;
- Возраст;
- Симптомы болезни;
- Диагноз;
- Отделение и палата, в которых находится пациент;
- ФИО лечащего врача;
- Название и дозировка препаратов, которыми будет производится лечение.

На рисунке 3 представлена структура функциональной связи отдела медстатистики с другими подразделениями больницы при поступлении информации о выбывших больных.

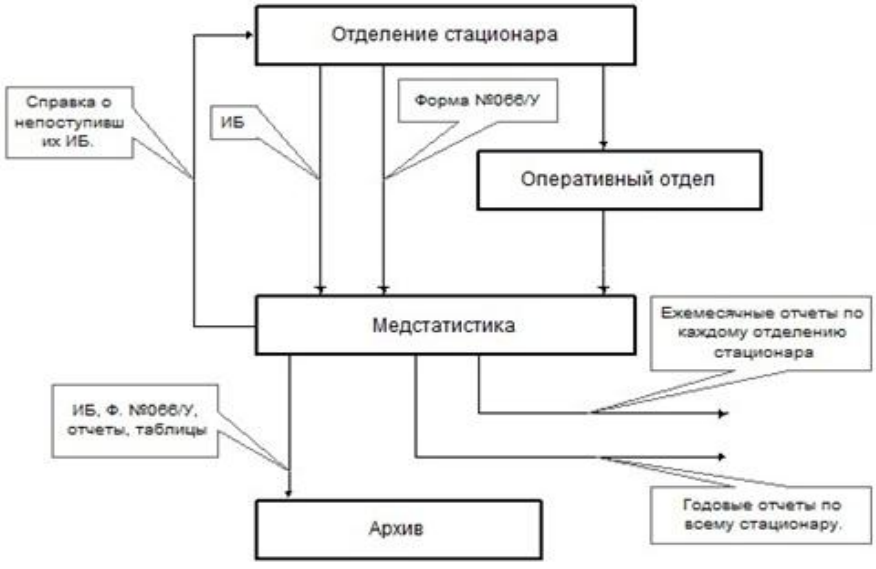

Рисунок 5 - Технологическая схема работы отдела медстатистики

1.4 Требования к разрабатываемой системе

Требования к программе:

Программа должна выполнять функции ввода, хранения и обработки информации о выбывшем больном и формирования ежедневных, месячных и годовых отчетов по фиксированным формам.

В программе должен быть обеспечен:

- ввод непротиворечивой и целостной информации;
- удобство пользователя, не имеющего специальных навыков работы с вычислительной техникой, высокая скорость заполнения за счет повышения наглядности и упрощения процесса ввода отдельных элементов информации;
- получение фиксированных отчетных форм за время, приемлемое для пользователя (не более 5 минут для ежедневного отчета, не более 1 часа для месячного отчета и не более 1 рабочего дня для годового отчета).

Контроль непротиворечивости и целостности информации должен состоять из двух этапов:

1. Контроля непосредственно при вводе (проверка ограничений на элемент информации, вводимый в данный момент);
2. Контроля информации после завершения ввода (проверка ограничений на совокупность элементов информации).

Для удобства пользователя в качестве базы для экранного интерфейса необходимо использовать полноэкранное редактирование полей БД.

Входной информацией программы является информация поступившем в стационар пациенте.

Выходной информацией программы являются таблицы:

- месячного отчета;
- годового отчета, включающие в себя:
  - Состав больных в стационаре, сроки и исход лечения;
  - Хирургическая работа учреждений;
  - Распределение больных по возрасту и направлению лечения;

- Больные, переведенные в другие лечебные учреждения;
- Нозология больных, переведенных из других стационаров;
- Нозология инфекционных заболеваний;
- Распределение инфекционных заболеваний в профильные стационары;
- Распределение выбывших иногородних больных по каналам госпитализации и отделениям больницы;
- Число больных, переведенных в другие стационары, из них число новорожденных, переведенных в другие стационары, и число лиц, госпитализированных для обследования и оказавшихся здоровыми.

Таблицы формируются для написания отчетов о деятельности стационара для администрации больницы и в вышестоящие организации, а также только для администрации больницы.

Взаимодействие программы с пользователем должно быть реализовано по следующему алгоритму:

1. Ввод текущей даты.
2. Выбор из главного меню одной из следующих функций:

2.1. Ввод данных и печать;

2.2. Коррекция ранее введенных данных;

2.3. Поиск данных больного по №ИБ или по Фамилии И.О;

2.4. Получение отчетных форм (за отчетный период с <число, месяц> по <число, месяц>);

2.5. Выход из программы.

После выполнения каждой функции необходимо обеспечить возврат в главное меню и выбор любой другой функции. При выборе функции "Выход из программы" работа программы завершается.

При выборе функции "Коррекция ранее введенных данных" пользователь может откорректировать данные, вводимые ранее.

Предусмотрен также режим "Настройка программы", который должен обеспечивать адаптацию программы к изменяющимся справочникам, используемым в программе, а именно:

- Добавление и удаление списка отделений стационара;
- Добавление и удаление списка профилей коек;
- Добавление и удаление списка направляющих организаций;
- Добавление и удаление списка операций;
- Добавление и удаление списка районов;
- Добавление и удаление списка возрастов;
- Добавление и удаление списка учреждений, откуда и куда переводятся больные;
- Добавление и удаление причин перевода больного;
- Добавление и удаление причин направлений больного;
- Добавление и удаление стран содружеств и их шифров.

Режим настройки должен быть скрыт от конечного пользователя программы и доступен лишь администратору БД.

Требования к надежности:

Разработанная программа должна обладать средствами защиты от ошибочных действий персонала.

Для предотвращения некорректной работы программы необходимо реализовать:

- семантический и синтаксический контроль исходных данных;
- вывод сообщений об ошибках;
- возможность повторного ввода.

Условия эксплуатации:

Программа должна быть ориентирована на пользователя, не имеющего специальных навыков работы в области вычислительной техники и программирования.

Требования к информационной и программной совместимости:

Данная программа должна представлять собой самостоятельный исполняемый модуль.

Информационная система должна быть реализована с использованием одного из известных языков программирования (C++, Oracle или др.) и работать под управлением операционной системы Windows.

Для разрабатываемого программного обеспечения должна быть разработана следующая техническая документация:

- текст программы;
- программа и методика испытаний;
- руководство оператора;
- руководство системного программиста.

ГЛАВА 2 Проектирование системы

2.1 Стадии и этапы разработки

Создание и внедрение системы включает следующие этапы:

– обследование организации, а также его программно-технического комплекса;

– организация проекта создания и внедрение автоматизированной системы учета прибывших и убывших из стационара больных;

– проведение обучения участников проекта от заказчика.

В таблице 1 представлены этапы проектирования АИС и их характеристики.

Таблица 1 – Этапы проектирования АИС

№ п/п	Наименование этапа	Основные характеристики
1.	Разработка и анализ бизнес - модели	Определяются основные задачи АИС, проводится декомпозиция задач по модулям и определяются функции с помощью которых решаются эти задачи. Описание функций осуществляется на языке производственных (описание процессов предметной области), функциональных (описание форм обрабатываемых документов) и технических требований (аппаратное, программное, лингвистическое обеспечение АИС). **Метод решения**: Функциональное моделирование. **Результат**: 1.Концептуальная модель АИС, состоящая из описания предметной области, ресурсов и потоков данных, перечень требований и ограничений к технической реализации АИС. 2.Аппаратно-технический состав создаваемой АИС.
2.	Формализация	Разработанная концептуальная модель

	бизнес - модели, разработка логической модели бизнес - процессов.	формализуется, то есть воплощается в виде логической модели АИС. **Метод решения**: Разработка диаграммы "сущность-связь" (ER (Entity-Reationship) - CASE- диаграммы). **Результат:** Разработанное информационное обеспечение АИС: схемы и структуры данных для всех уровней модульности АИС, документация по логической структуре АИС, сгенерированные скрипты для создания объектов БД.
3.	Выбор лингвистического обеспечения, разработка программного обеспечения АИС.	Разработка АИС: выбирается лингвистическое обеспечение (среда разработки - инструментарий), проводится разработка программного и методического обеспечения. Разработанная на втором этапе логическая схема воплощается в реальные объекты, при этом логические схемы реализуются в виде объектов базы данных, а функциональные схемы - в пользовательские формы и приложения. **Метод решения:** Разработка программного кода с использованием выбранного инструментария. **Результат**: Работоспособная АИС.
4.	Тестирование и отладка АИС.	На данном этапе осуществляется корректировка информационного, аппаратного, программного обеспечения, проводится разработка методического обеспечения (документации разработчика, пользователя) и т.п. **Результат**: Оптимальный состав и эффективное функционирование АИС. Комплект документации: разработчика,

		администратора, пользователя.
5.	Эксплуатация и контроль версий	Особенность АИС созданных по архитектуре клиент сервер является их многоуровневость и многомодульность, поэтому при их эксплуатации и развитии на первое место выходят вопросы контроля версий, то есть добавление новых и развитие старых модулей с выводом из эксплуатации старых. Например, если ежедневный контроль версий не ведется, то в как показала практика, БД АИС за год эксплуатации может насчитывать более 1000 таблиц, из которых эффективно использоваться будет лишь 20-30%. **Результат**: Наращиваемость и безизбыточный состав гибкой, масштабируемой АИС

Процесс работы с системой разбивается на несколько этапов:

- ввод текущей даты;
- работа в главном меню;
- ввод новых данных;
- коррекция введенных данных;
- формирование и печать отчетов за выбранный период;
- настройка программы на структуру конкретного стационара;
- выход из программы.

На рисунке 6 представлен процесс работы с системой поэтапно.

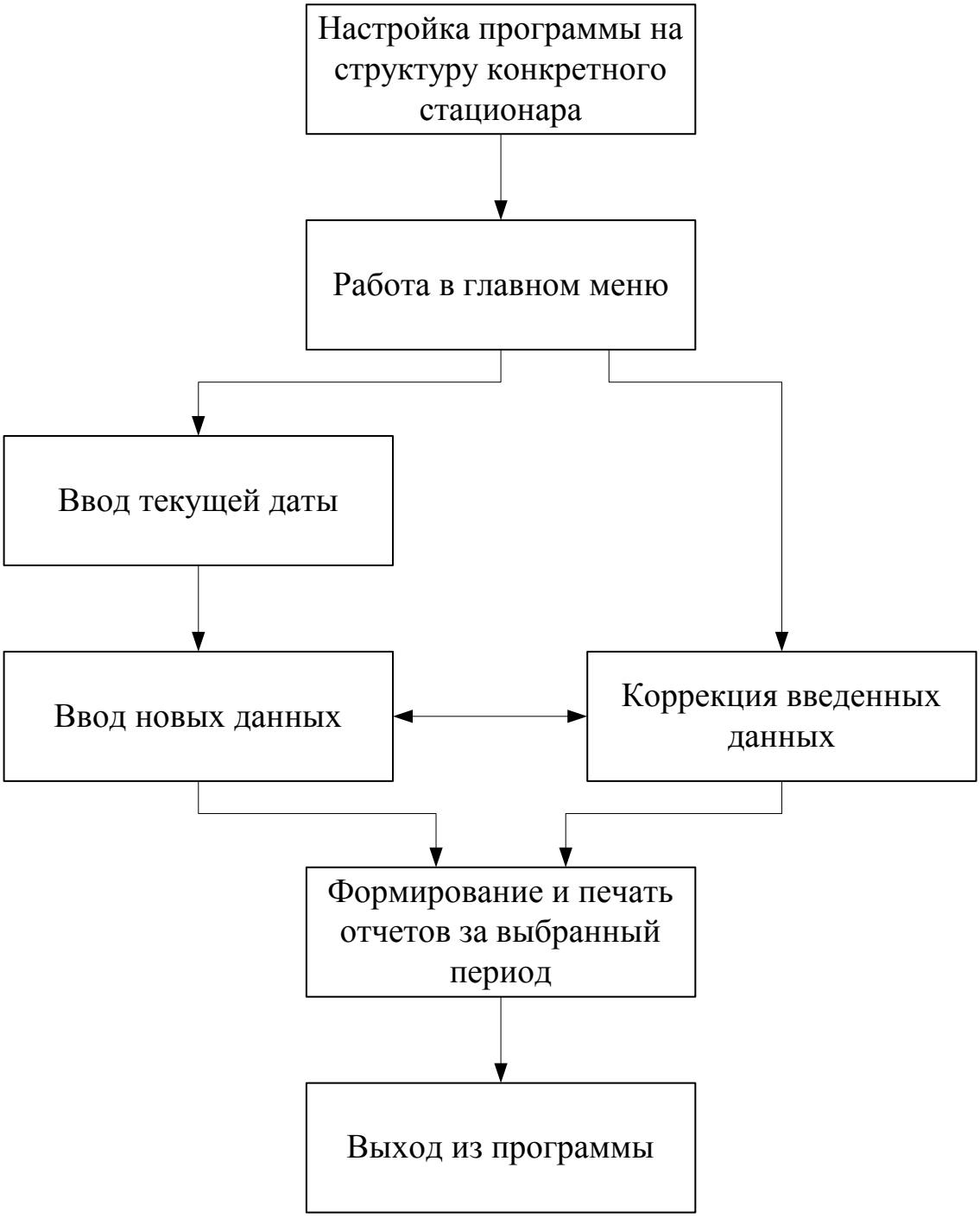

Рисунок 6 - Процесс работы с системой

Входные документы

Источниками информации для разработки системы будет являться ряд медицинских справок:

- справка о госпитализации;
- направление на прием к врачу;
- заключение врача;
- медицинская карта;

- результаты анализов;
- справка о выписке;
- протоколы консилиумов.

Рисунок 7 - Источники информации для разработки системы.

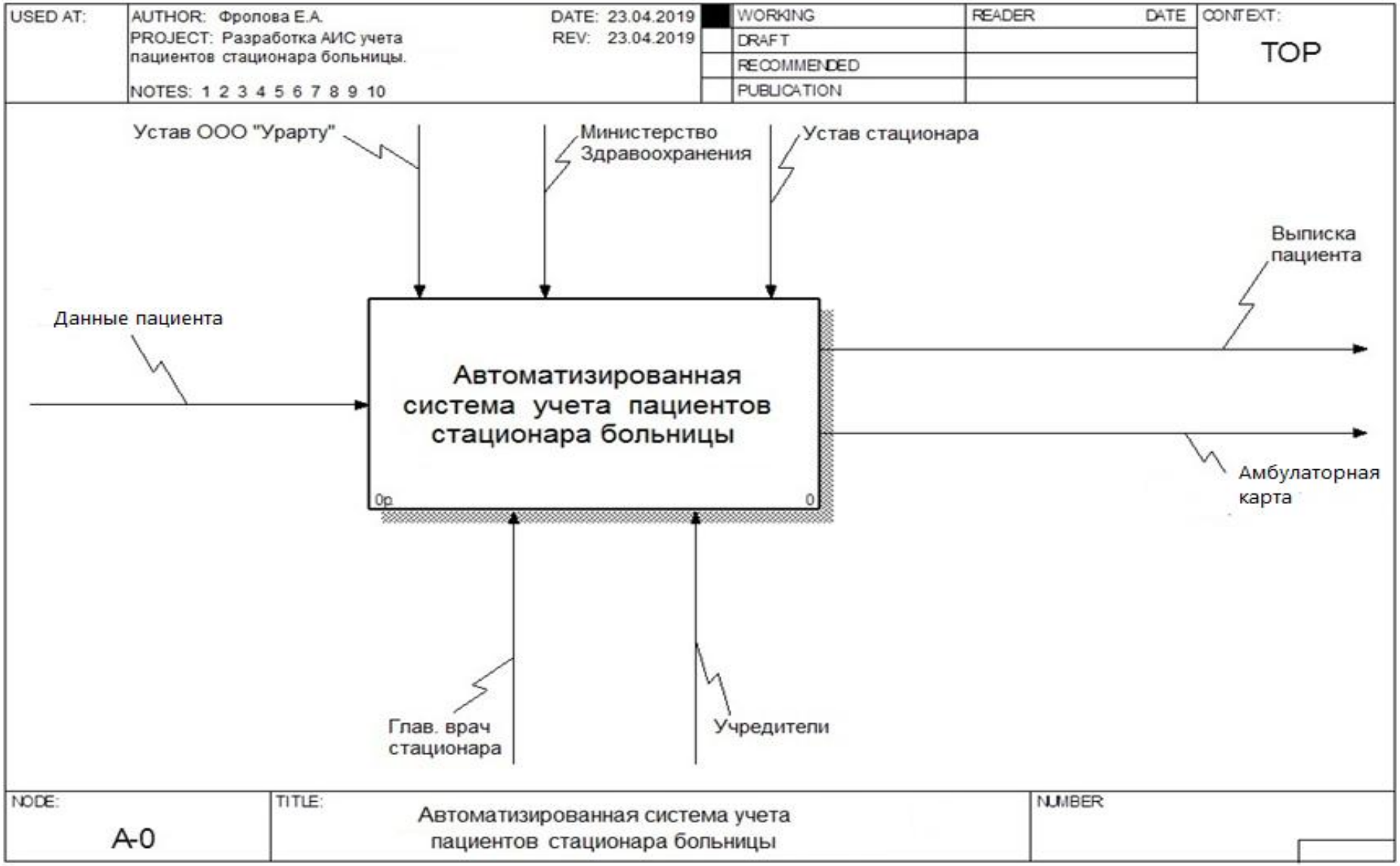

Рисунок 8 - Схема работы после автоматизации

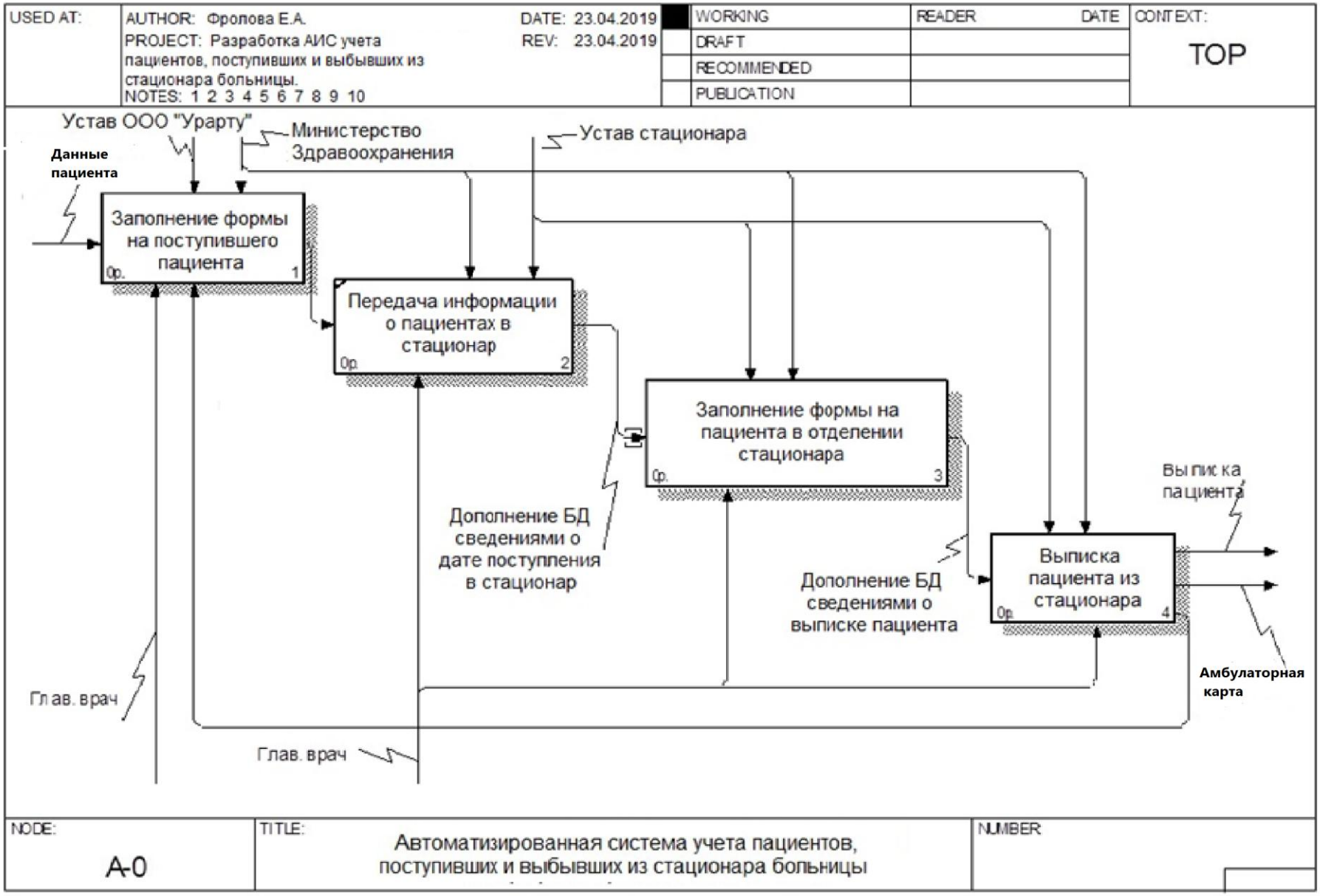

Рисунок 9 - Схема работы после автоматизации

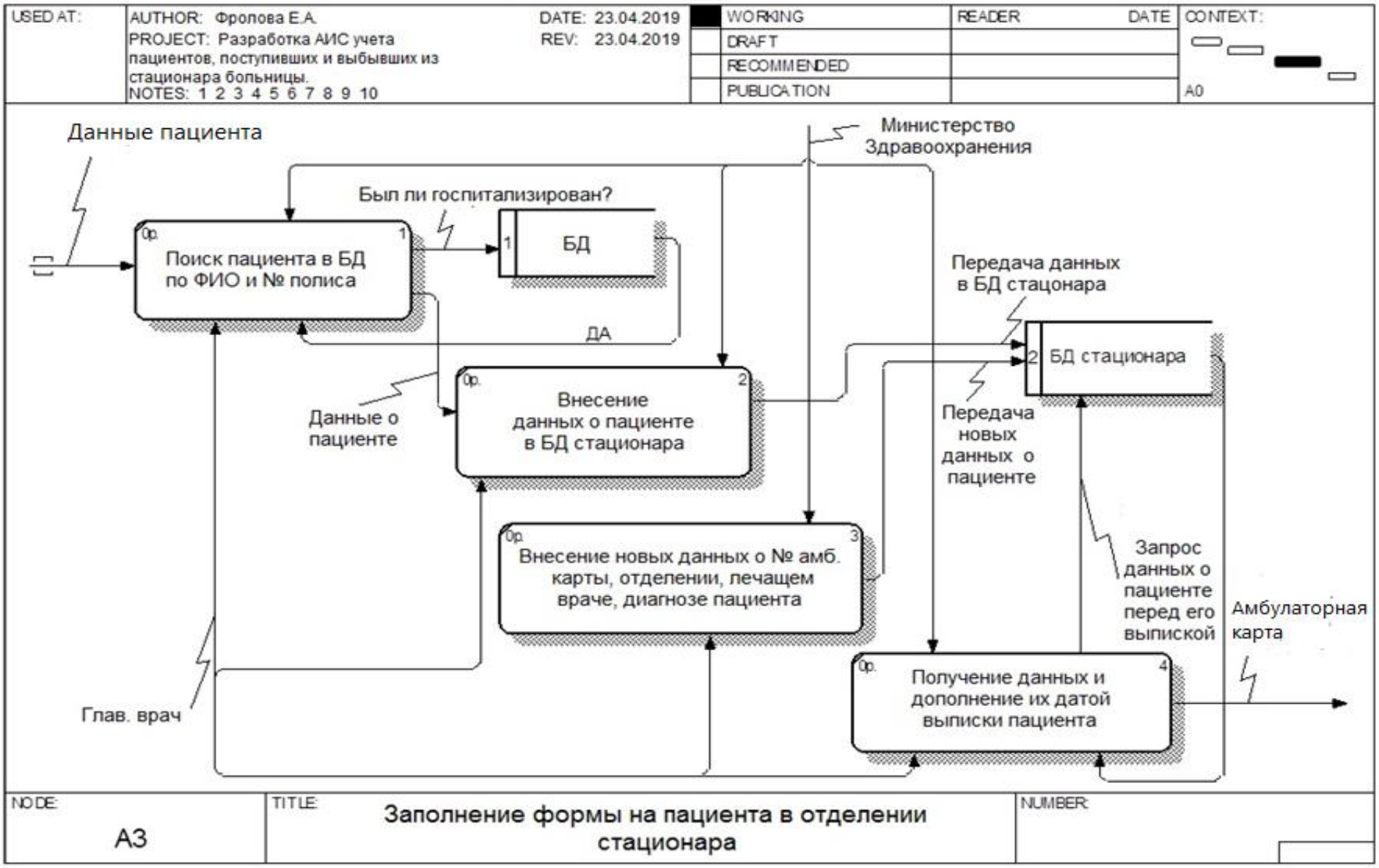

Рисунок 10 - Схема работы после автоматизации

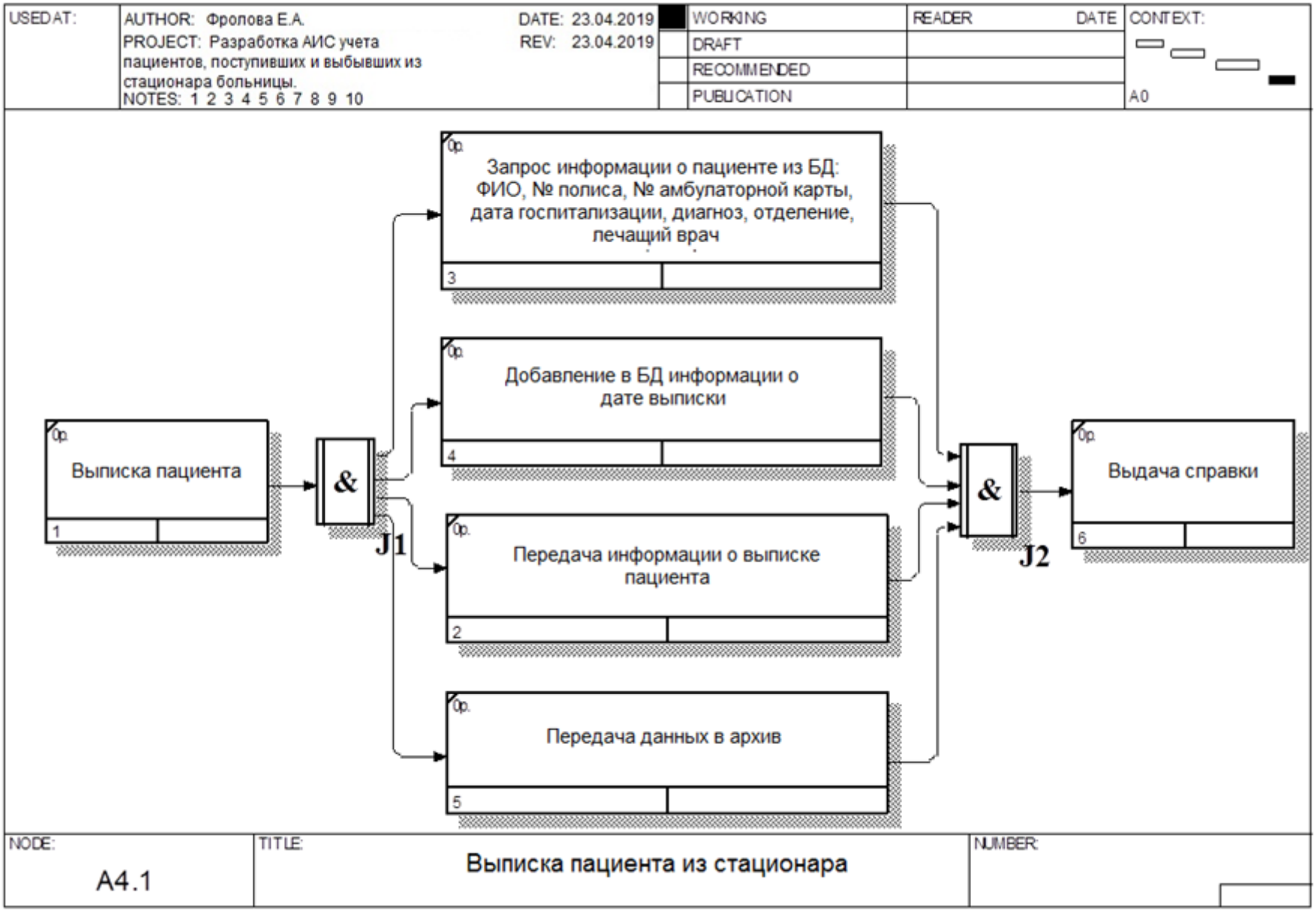

Рисунок 11 - Схема работы после автоматизации

2.2 Построение логической модели

Логическая модель является визуальным представлением структур данных, их атрибутов и бизнес-правил. Логическая модель представляет данные таким образом, чтобы они легко воспринимались пользователями. Проектирование логической модели должно быть свободно от требований платформы и языка реализации или способа дальнейшего использования данных.

Разработчик модели использует требования к данным и результаты анализа для формирования логической модели данных.

Компоненты логической модели данных

Логическая модель использует сущности, атрибуты и отношения для представления данных и бизнес-правил. ***Сущности*** представляют собой объекты, о которых корпорация заинтересована хранить данные. ***Атрибуты*** - это данные, которые корпорация заинтересована хранить. ***Отношения*** описывают взаимосвязи между сущностями в терминах бизнес-правил.

Сущности представляют собой объекты, данные о которых корпорация заинтересована сохранять. Сущностями являются вещественные объекты, такие как пациент, но они могут представлять и абстрактные концепции, такие как выписка.

Каждый экземпляр сущности должен быть отдельным и отличным от всех других экземпляров этой сущности.

Атрибуты представляют данные об объектах, которые необходимо иметь корпорации. Атрибуты представляются именами существительными, которые описывают характеристики сущностей.

Отношения представляют взаимосвязи между объектами, о которых корпорация заинтересована хранить данные. **Отношения** выражаются глаголами или глагольными фразами, которые описывают взаимосвязь.

Завершенная логическая модель данных

Завершенная логическая модель должна удовлетворять требованиям третьей бизнес-нормальной формы и включать все сущности, атрибуты и связи, необходимые для поддержки требований к данным.

Все сущности должны иметь имена, описывающие содержание и ясное, краткое, полное описание или определение.

Сущности должны иметь полный набор атрибутов, так, чтобы каждый факт относительно каждой сущности мог быть представлен ее атрибутами. Каждый атрибут должен иметь имя, отражающее его значения, логический тип данных и ясное, короткое, полное описание или определение.

Связи должны включать глагольную конструкцию, которая описывает отношение между сущностями, наравне с такими характеристиками, как множественность, необходимость существования или возможность отсутствия связи.

На рисунке 12 представлена логическая модель данных.

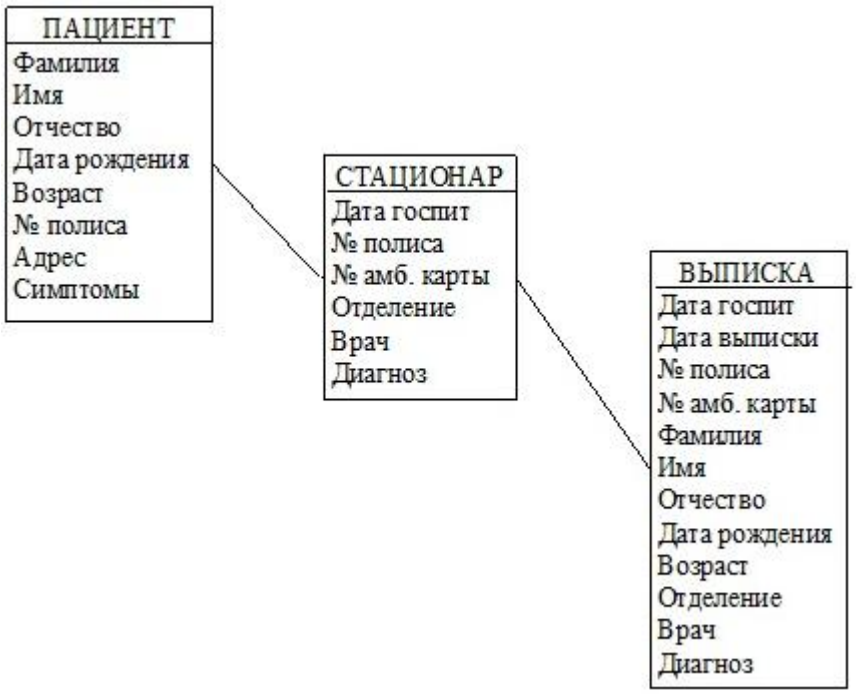

Рисунок 12 - Логическая модель данных

2.3 Построение физической модели

После создания полной и адекватной логической необходимо принять решения о выборе платформы реализации. Выбор платформы зависит от

требований к использованию данных и стратегических принципов формирования архитектуры корпорации.

В ERwin физическая модель является графическим представлением реально реализованной базы данных. Физическая база данных будет состоять из таблиц, столбцов и связей. Физическая модель зависит от платформы, выбранной для реализации, и требований к использованию данных.

Можное денормализовать физическую модель для улучшения производительности, и создать представления для поддержки требований к использованию.

Компоненты физической модели данных

Компонентами физической модели данных являются таблицы, столбцы и отношения. Сущности логической модели, вероятно, станут таблицами в физической модели. Логические атрибуты станут столбцами. Логические отношения станут ограничениями целостности связей. Некоторые логические отношения невозможно реализовать в физической базе данных.

На рисунке 13 представлена физическая модель данных.

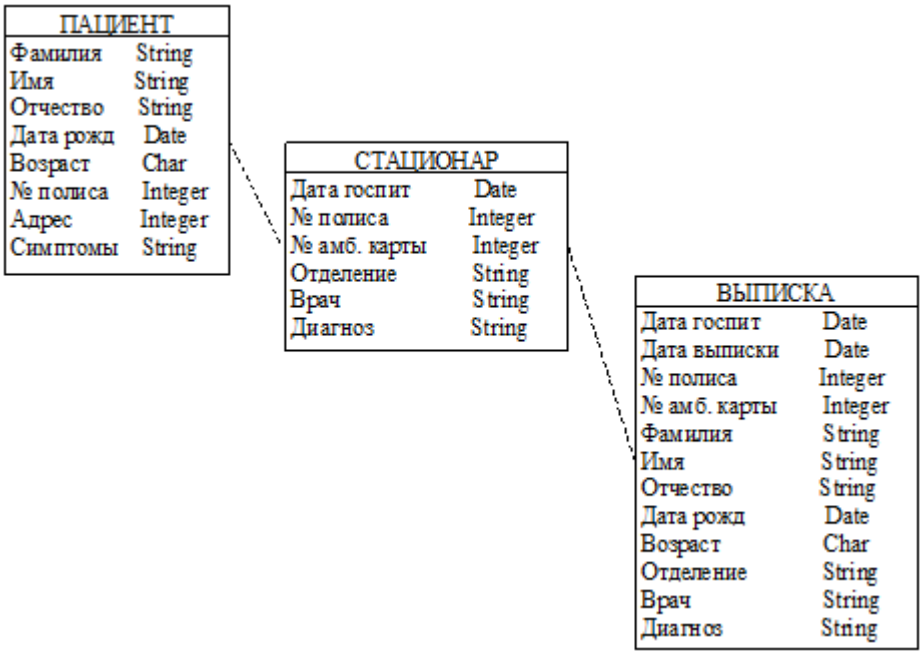

Рисунок 13 - Физическая модель данных

ГЛАВА 3 Техническая реализация системы

3.1 Технология реализации

Для разработки программы использован объектно-ориентированный язык программирования C++..

C++ — компилируемый, статически типизированный язык программирования общего назначения. **C++** широко используется для разработки программного обеспечения, являясь одним из самых популярных языков программирования. Область его применения включает создание операционных систем, разнообразных прикладных программ, драйверов устройств, приложений для встраиваемых систем, высокопроизводительных серверов.

Основной идеей объектно ориентированного подхода является объединение данных и действий, производимых над этими данными, единой целое, которое называется объектом. Типичная программа на языке C++ состоит из совокупности объектов, взаимодействующих между собой по средствам вызова методов друг друга.

C++ Builder это инструмент для быстрой разработки приложений и система, используемая программистами для разработки программного обеспечения на языке программирования C++.

C++ Builder содержит инструменты, которые при помощи drag-and-drop делают разработку визуальной, упрощает программирование

Программа позволяет осуществлять контроль за разграничекнием прав доступа к данным и работе с ними. Самые раширенные права имеет администратор.Доступ к файлам и данным по истории болезни пациента имеют лечащий врач и мед сестры. Ниже представлен фрагмент кода данного раздела программы.

```
//заполнение таблицы документов
procedure TfmMain.DocFill();
var sql, where_p, inner_p: string;
   keyword: string;
```

```
begin

 //учёт прав доступа на документ у всех кроме админа
 inner_p:= '';
 if CurrentUserType <> 1 then
  inner_p:= 'INNER JOIN DocPerm DP ON (DP.USER_ID=' + IntToStr(CurrentUserID) + '
AND DP.DOCUMENT_ID=D.ID)';

 //поиск по ключевому слову
 keyword:= AnsiUpperCase(leFind.Text);

 where_p:= '';
 if keyword <> '' then where_p:= ' AND upper(D.NAME) LIKE '#39'%' + keyword + '%'#39;

 //собираем запрос
 sql:= 'SELECT D.* FROM Document D ' + inner_p + '  WHERE D.CATEGORY_ID=' +
IntToStr(CurCateg) + where_p + ' ORDER BY D.NAME';

 sqlDoc.SQL.Clear;
 sqlDoc.SQL.Add(sql);
 sqlDoc.Open;
 sqlDoc.FetchAll;

 if sqlDoc.RecordCount = 0 then
 begin
  CurDoc:= -1;

  btEdit.Enabled:= false;
  btDel.Enabled:= false;
  btLoad.Enabled:= false;
  btRead.Enabled:= false;
  btDocArhiv.Enabled:= false;

  mDescription.Clear;

  exit;
 end;

 if not sqlDoc.Locate('Id', CurDoc, []) then  CurDoc:= sqlDocId.Value;


 if fmS.DocStatus(sqlDocState.Value) <> 0 then btLoad.Caption:= 'Сохранить'
 else btLoad.Caption:= 'Редактировать';

 mDescription.Text:= sqlDocDescription.Value;

 btEdit.Enabled:= true;
 btDel.Enabled:= true;
 btLoad.Enabled:= true;
 btRead.Enabled:= true;

 btDocArhiv.Enabled:= fmS.IsArhiv(CurDoc);
```

```
end;
```

Данная функция осуществляется при запуске программы и во время её настройки под роль отдельного пользователя. Листинг загрузки параметров базы данных :

```
procedure SaveCFG(); //сохраняем параметры
begin
 AssignFile(CFG_File, Dir + 'config.cfg');
 Rewrite(CFG_File);

 SaveDBConfig(); //настройки БД

 CloseFile(CFG_File);
end;

//---------------------------------------------------------------------------

procedure LoadDBConfig(); //загружаем настройки БД
var s: string;
begin
 readln(CFG_File,s);
 while (s <> '[db]') and not Eof(CFG_File) do readln(CFG_File,s);
 with CFG.DBConfig do
 begin
  readln(CFG_File,ServerName);
  readln(CFG_File,UserName);
  readln(CFG_File,Password);
 end;
end;

//---------------------------------------------------------------------------

procedure LoadCFG(); //загружаем параметры
begin
 if not FileExists(Dir + 'config.cfg') then
 begin
  ClrCFG();
  exit;
 end;

 AssignFile(CFG_File, Dir + 'config.cfg');
 Reset(CFG_File);

 LoadDBConfig(); //настройки БД

 CloseFile(CFG_File);
end;

//---------------------------------------------------------------------------
```

```
    //очистка cfg файла
procedure ClrCFG();
begin
 //настройки БД
 with CFG.DBConfig do
 begin
  ServerName:='';
  Username:='';
  Password:='';
 end;
end;

//---------------------------------------------------------------------------

initialization
 CurrentUserID:= 0;
 CurrentUserType:= 0;

end.
```

Есть возможность сохранять лог ошибок, работать с большими массивами данных, сохранять отчеты во всех современных текстовых файлах.

3.2 Экранные формы

Данная программа предназначена для учета пациентов стационара вследствие госпитализации.

Работа в автоматизированной информационной системе начинается с выбора системы, в которой предстоит работать. Если в программе работает оператор ЭВМ стационара или медицинский персонал, то ему необходимо войти в систему «Стационар».

При нажатии кнопки «Информация» пользователь может узнать для чего предназначена эта система, а так же про ее разработчика.

При выборе кнопке «Выход» будет осуществлен выход из программы.

На рисунке 15 представлен вход в автоматизированную больничную информационную систему.

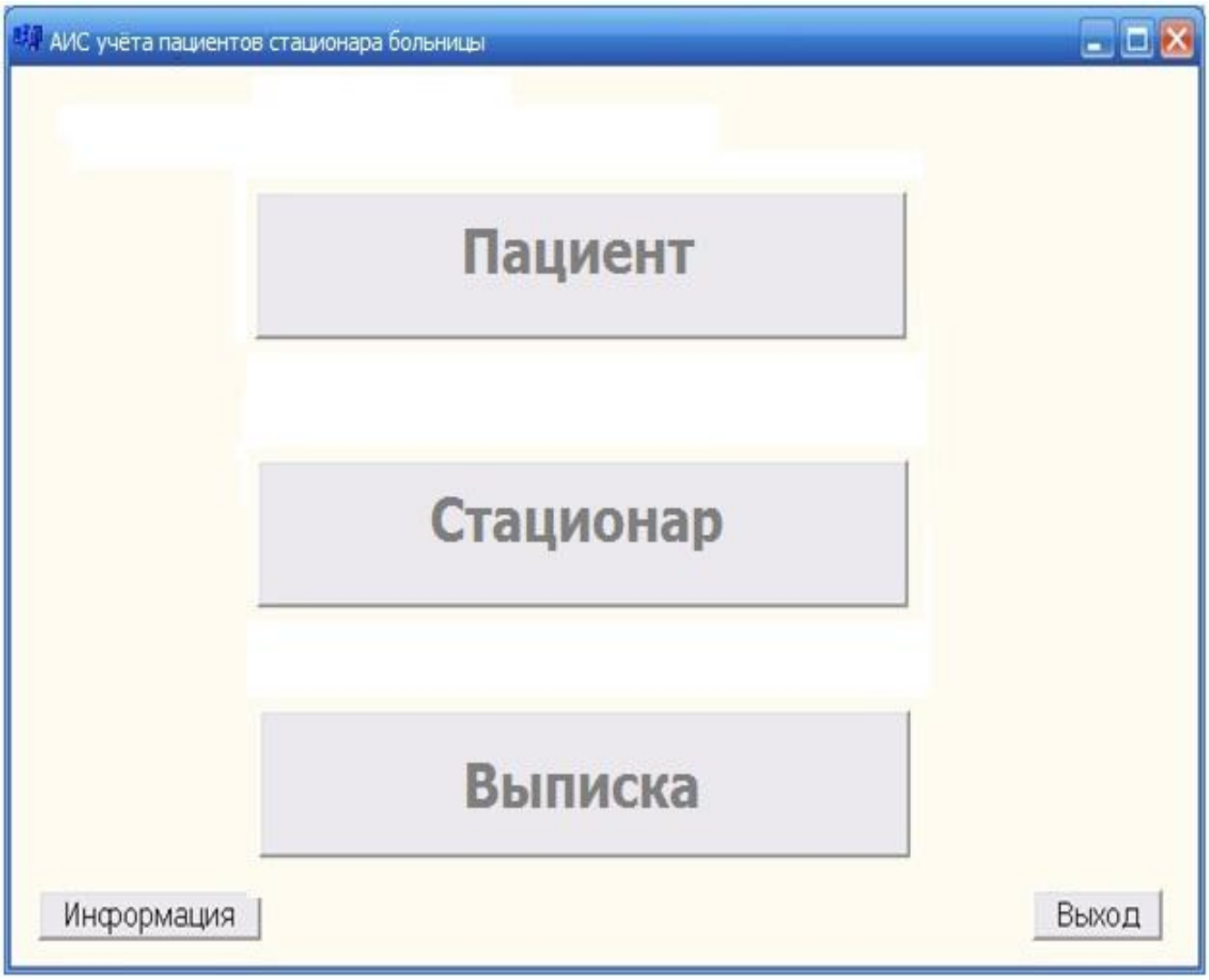

Рисунок 15 - Вход в АИС учёта пациентов стационара больницы

В системе организован поиск пациентов по № полиса и по фамилии. Если пациент не внесен в базу, то пользователю будет выведено сообщение о том, что пациент не найден.

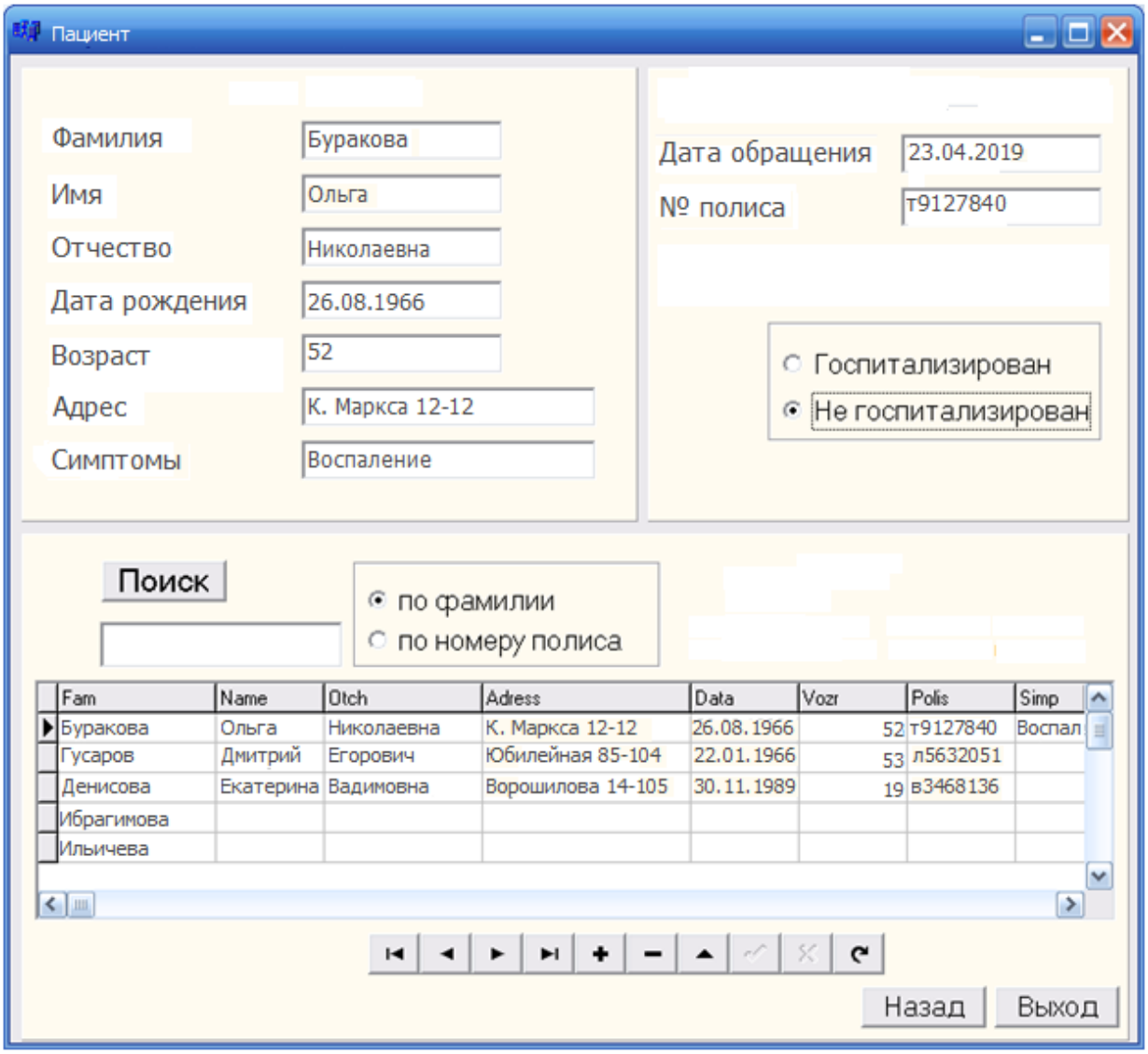

Рисунок 16 - Работа в окне «Пациент»

На следующем рисунке (рисунок 17) представлена работа в окне «Стационар». Переданная информация по каждому госпитализированному пациенту вносится в базу данных стационара, но в эти данные добавляются еще сведения об отделении, в котором пациент находится, о его лечащем враче, а также № его амбулаторной карты.

Для удобства так же можно просматривать всю базу данных, добавлять, удалять и корректировать в ней данные.

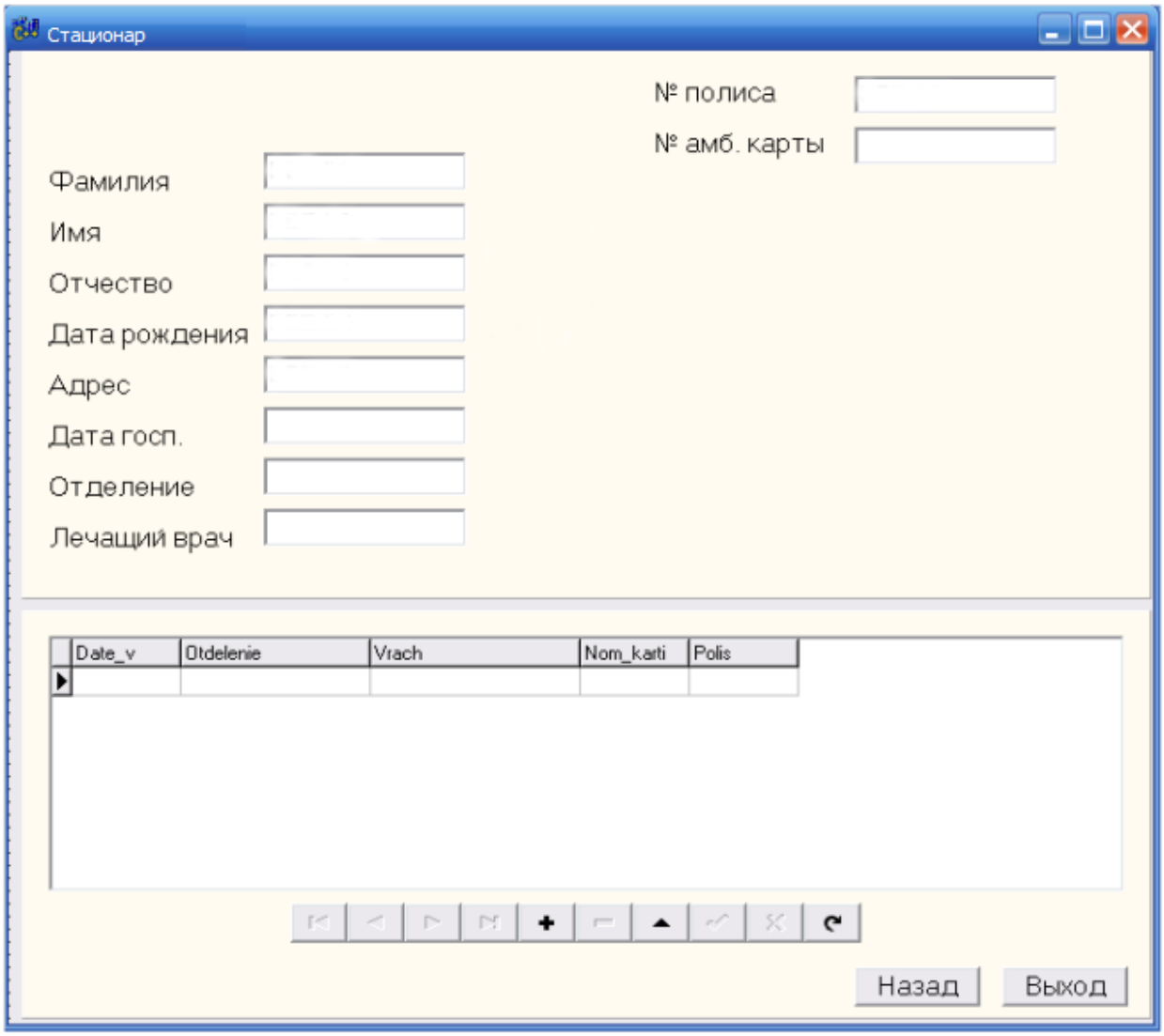

Рисунок 17 - Работа в окне «Стационар»

На рисунке 18 представлена работа в окне «Выписка». Данная форма заполняется также операторами ЭВМ стационара, но просматриваться может как работниками стационара, так и врачами скорой медицинской помощи.

Работа в этом окне осуществляется с помощью поиска пациентов либо по фамилии, либо по № полиса, либо по № амбулаторной карты.

Также здесь появляется новое поле о дате выписки пациента.

Данные этой системы могут быть направлены на печать справки о выписке из стационара либо на хранение в архив.

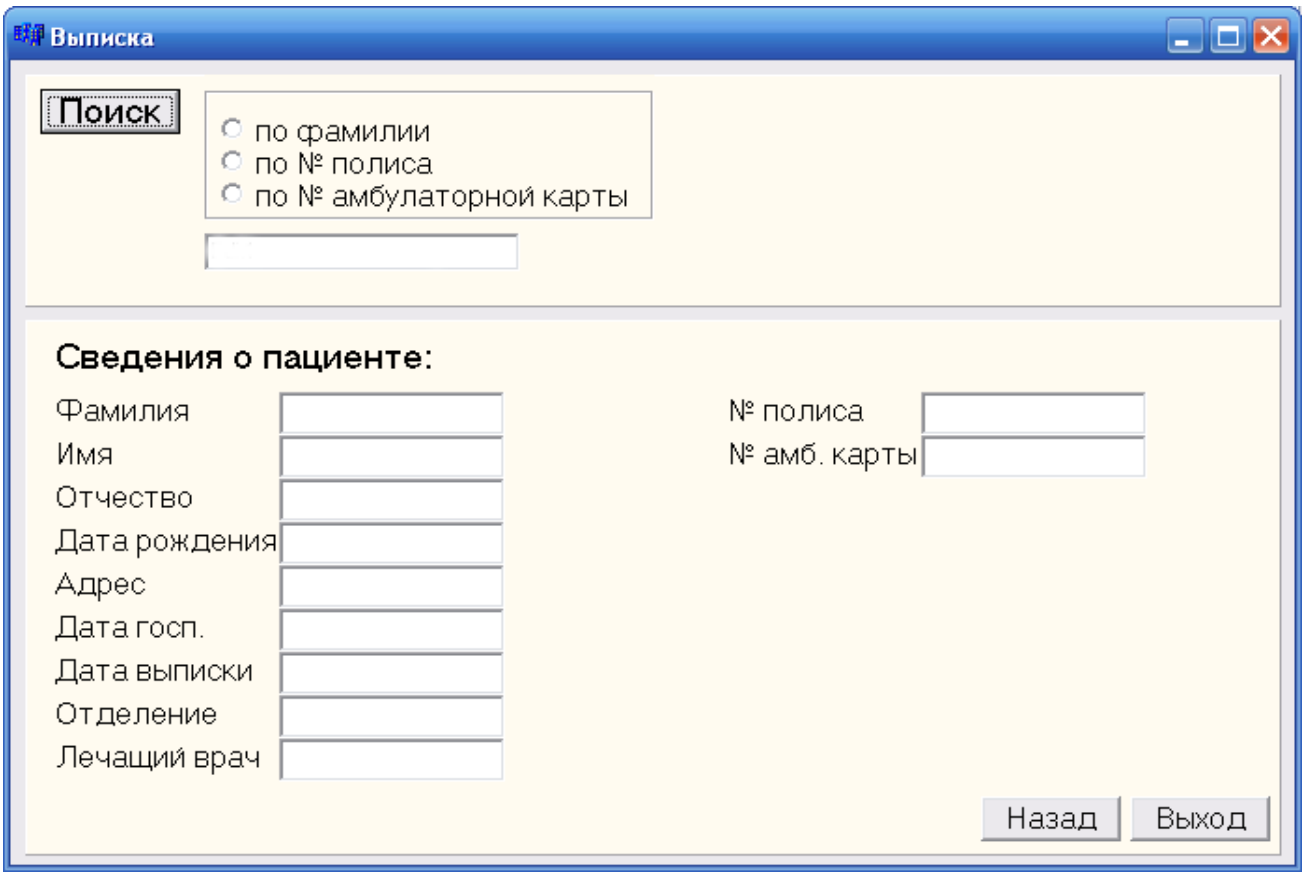

Рисунок 18. Работа в окне «Выписка»

На рисунках 16, 17 и 18 также присутствуют кнопки «Назад» и «Выход». С помощью кнопки «Назад» осуществляется переход в самое начало системы, где осуществляется выбор, в какой системе работать. Кнопка «Выход» - выход из программы.

3.3 Расчет затрат на разработку системы учета пациентов стационара

Расчет затрат на разработку необходим для обоснования экономической эффективности системы. Плановые затраты на разработку включают все расходы, связанные с ее выполнением, независимо от источника их финансирования. Определение затрат на разработку производится путем составление калькуляции плановой себестоимости. Она является основным документом, на основании которого осуществляется планирование и учет затрат на выполнение.

Смета затрат включает следующие статьи:

– основная заработная плата разработчиков информационной системы;

– дополнительная заработная плата разработчиков информационной системы;

– отчисления на социальные страхования;

– расчет затрат на амортизацию ЭВМ;

– расходы на электроэнергию, используемую на разработку информационной системы;

– накладные расходы.

При расчете затрат на создание системы был составлен график стадий разработки и их продолжительность. Перечень работ по созданию АИС приведен в таблице 2.

Таблица 2 - Перечень работ по созданию АИС

№ п/п	Наименование этапа	Продолжительность, дней
1	Анализ предметной области	4
2	Разработка технического задания	5
3	Выбор комплекса технических средств	5
4	Разработка логической модели	5
5	Разработка алгоритмов	38
6	Разработка интерфейса	8
7	Тестирование и отладка ПО	5
Итого:		70

Используя данные из таблицы, сформируем график разработки АИС (рисунок 19). Вся разработка выполняется одним работником. Данные о продолжительности работы необходимы для расчета затрат на разработку автоматизированной системы учёта пациентов стационара больницы.

Рисунок 19 - График разработки автоматизированной системы учёта пациентов стационара больницы

На рисунке 20 представлена сравнительная диаграмма экономической эффективности.

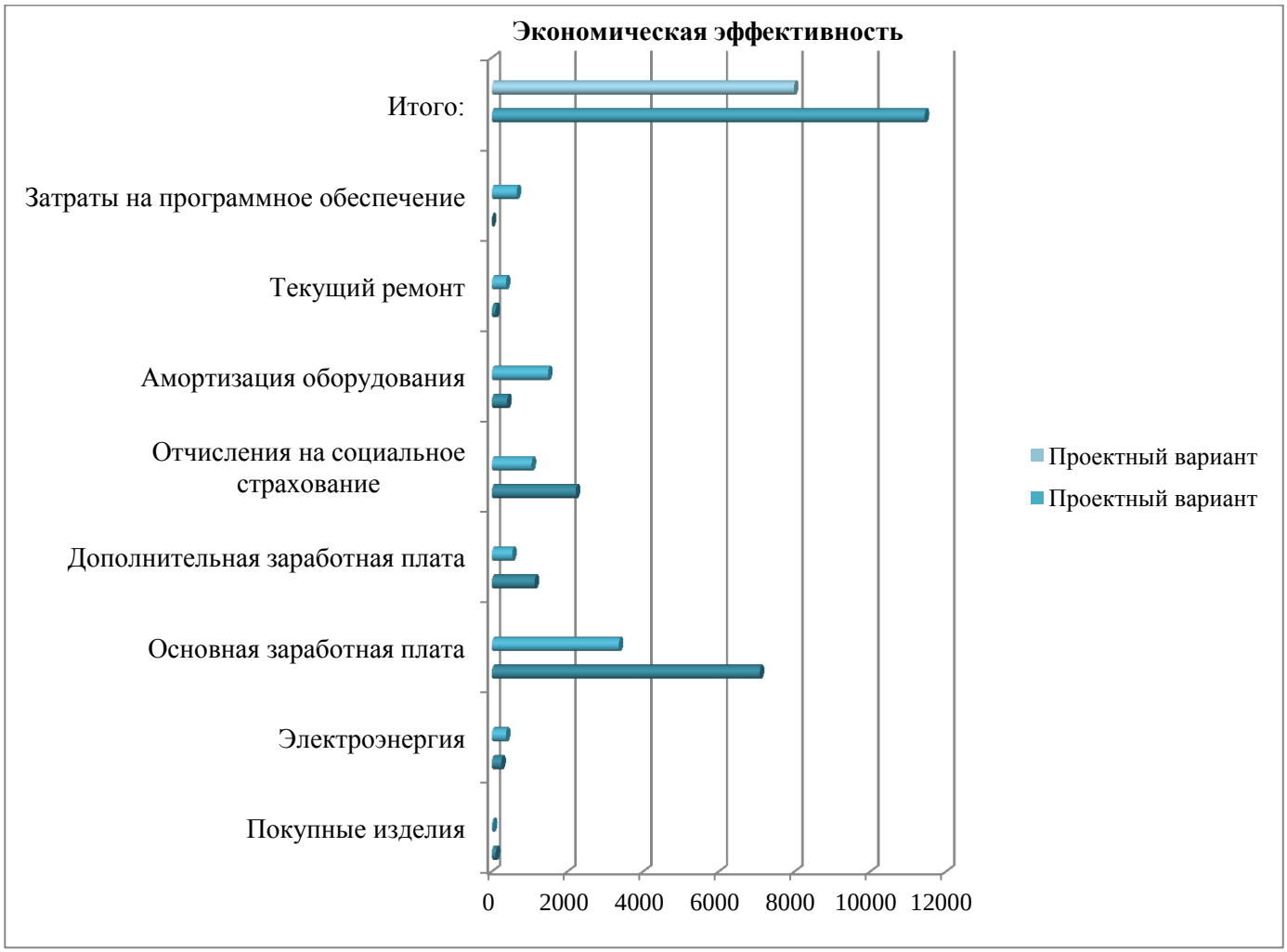

Рисунок 20 - Сравнительная диаграмма экономической эффективности

Экономический анализ показал, что разработанную автоматизированную систему учёта пациентов из стационара больницы целесообразно использовать для оптимизации рабочих процессов.

ЗАКЛЮЧЕНИЕ

В результате проделанной работы были получены следующие результаты:

– проведен анализ существующих методов создания информационных систем в медицине;

– проведен анализ работы отдела медстатистики стационара и анализ потоков информации, проходящей через него, на основании чего сделан вывод о возможности использования в работе отдела стандартной сети Ethernet, с пропускной способностью 10 Мбит/с;

– в качестве программных средств – язык программирования C++Builder;

– составлено формализованное описание документооборота;

– разработана концептуальная модель базы данных, содержащая в себе информацию и состоящую из трех баз данных, связанных по номеру медицинского полиса;

– разработаны алгоритмы обработки;

– разработаны организационно - технические мероприятия, направленные на поддержание сохранности и целостности БД;

– определен годовой экономический эффект от внедрения разработанной автоматизированной системы.

Система предназначена для работы в отделениях медстатистики больниц для учета выбывших из стационаров и не требует от пользователя дополнительных знаний по программированию, она предоставляет ему удобный экранный интерфейс для работы с многофайловой базой данных.

Система прошла опытную эксплуатацию в стационаре ООО «Урарту».

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Авинаш, К. WebAnalytics 2.0: The Art of Online Accountability and Science of Customer Centricity. / К.Авинаш, Sybex, 2014.
2. Александров, Д.В. Инструментальные средства информационного менеджмента. CASE-технологии и распределённые информационные системы: учебное пособие /Д. В.Александров. – МФиС, 2011.
3. Баронов В. В., «Автоматизация управления предприятием», Москва, Инфра-М, 2000 г.
4. Комментарий к Трудовому кодексу РФ, постатейный, научно-практический/ под. ред. К. Я. Ананьевой, М.:ТОН-ЮРАЙТ, 2002 г.;
5. В.А. Виттих, Д.В. Волхонцев, Н.И. Кантемирова, О.П. Ляпина, И.М. Назаркина,П.О. Скобелев, О.Л. Сурнин, Л.С. Федосеева, М.А. Шамашов, И.С. Шапиро «Разработка автоматизированной информационной системы и экономической модели поликлиники, ориентированной на адресное оказание медицинских услуг населению», Институт проблем управления сложными системами РАН;
6. Мартин Дж. “Организация баз данных в вычислительных системах”: пер. с англ. - М:Мир, 1990. - 660с.:ил.;
7. Серия "Обзоры по важнейшим проблемам медицины": выпуск N 5: Использование автоматизированных больничных информационных систем за рубежом": М: Медицина и здравоохранение, 1998;
8. Дейт К.Дж.” Введение в системы баз данных”: Пер. с англ. - К: Диалектика,1998. - 784с.: ил;
9. “Гигиенические требования к персональным электронно-вычислительным машинам и организации работы” СанПин 2.2.2/2.4.1340-03;
10. Приказ Минздрава РФ “О санитарно-эпидемиологической экспертизе продукции” от 15.03.01г. № 325 (с изменениями от 18.03.02);
11. А.Я. Архангельский «Программирование в C++Builder 6»: M,

«Издательство БИНОМ», 2004г. – 1152 с.: ил.

12. Билл, С. Тереза, Н. Проектирование веб-интерфейсов. / Билл, С. Тереза, Н.- СПб. : Питер, 2010.

13. Бекет, Г. Мойду, М., Роди Ш. Тост А.Java ОсновоWeb-служб. / Г. Бекет, М. Мойду, Ш. Роди, А. Тост- СПб. : Питер, 2010.

14. Васильев, А. Н. Java.Объектно-ориентированное программирование: Учебное пособие / А. Н.Васильев,- СПб. : Питер, 201.

15. Гонсалвес, А. Э. Beginning Java EE 7 / Гонсалвес, А. Э.Apress 2014.

16. Избачков, Ю. С. Петров В.Н. Информационные системы / Ю. С. Избачков, В. Н. Петров - СПб. : Питер, 2011.

17. Кришнамурти, Б. Рексфорд Д. Web-протоколы. Теория и практика./ Б. Кришнамурти, Д. Рексфорд- СПб: Бином, 2012.

18. Марманис, Х. Бабенко, Д. Алгоритмы интеллектуального Интернета. Передовые. / Х. Марманис, Д. Бабенко,- СПб.: Питер, 2011.

19. Макконнелл, Совершенный код. / С.Макконнелл, СПб. : Питер, 2015.

20. Олифер, С. Компьютерные сети. Принципы, технологии, протоколы. / С.Олифер, СПб. : Питер, 2015.

21. Розенфельд, Л.Морвиль, П. Information Architectrre for the Word Wide Web / Л. Розенфельд, П. Морвиль, СПб.: Символ-Плюс, 2010.

22. Седжвик, Р. Уэйн К. Алгоритмы на Java, 4-е издание. / Р. Седжвик, К.Уэйн СПб. :Питер, 2013.

23. Уодтке, К. Информационная архитектура: Чертежи для сайта. / К.Уодтке, СПб.: Питер, 2010.

24. Уэзеролл, Т. Компьютерные сети. / Т.Уэзеролл, СПб. : Питер, 2014.

25. Флэнаган, Д. JavaScipt Полное руководство / Д.Флэнаган,- СПб.: Символ-Плюс, 2011.

26. Шкляр, Р. - Архитектура веб-приложений./ Р.Шкляр, СПб.: БХВ-Петербург, 2011.

27. Эккель, Б. Thinkingin Java / Б.Эккель, Prentice Hall, 2010

ПРИЛОЖЕНИЕ

Листинг программы

Модуль DocumentER.dpr

```
program DocumentER;

uses
  Forms,
  Main in 'Main.pas' {fmMain},
  SQLDM in 'SQLDM.pas' {fmS: TDataModule},
  Global in 'Global.pas',
  Avtorizate in 'Avtorizate.pas' {fmAvtorizate},
  AddDoc in 'AddDoc.pas' {fmAddDoc},
  About in 'About.pas' {fmAbout},
  AddUser in 'AddUser.pas' {fmAddUser},
  UserList in 'UserList.pas' {fmUserList},
  DocType in 'DocType.pas' {fmDocType},
  ActionPerm in 'ActionPerm.pas' {fmActionPerm},
  CategoryPerm in 'CategoryPerm.pas' {fmCategoryPerm},
  Log in 'Log.pas' {fmLog},
  Arhiv in 'Arhiv.pas' {fmArhiv};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TfmMain, fmMain);
  Application.CreateForm(TfmS, fmS);
  Application.CreateForm(TfmAvtorizate, fmAvtorizate);
  Application.CreateForm(TfmAddDoc, fmAddDoc);
  Application.CreateForm(TfmAbout, fmAbout);
  Application.CreateForm(TfmAddUser, fmAddUser);
  Application.CreateForm(TfmUserList, fmUserList);
  Application.CreateForm(TfmDocType, fmDocType);
  Application.CreateForm(TfmActionPerm, fmActionPerm);
  Application.CreateForm(TfmCategoryPerm, fmCategoryPerm);
  Application.CreateForm(TfmLog, fmLog);
  Application.CreateForm(TfmArhiv, fmArhiv);
  Application.Run;
end.
```

Модуль Main.pas

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Global, SQLDM, DbLogDlg, ExtCtrls, ComCtrls, Grids, DBGrids,
  StdCtrls, DB, IBDatabase, IBCustomDataSet, IBQuery, DateUtils, Menus, ShellApi,
  IBTable;

type

  TfmMain = class(TForm)
    Panel5: TPanel;
    Panel6: TPanel;
    tDoc: TIBTransaction;
    sqlDoc: TIBQuery;
    sqlDockalk_state: TStringField;
    dsDoc: TDataSource;
    mmMenu: TMainMenu;
    mmBase: TMenuItem;
    mmUserList: TMenuItem;
    mmClearUsers: TMenuItem;
    mmN2: TMenuItem;
    mmOwnData: TMenuItem;
    N1: TMenuItem;
    mmClose: TMenuItem;
    mmPerm: TMenuItem;
    mmActionsPerm: TMenuItem;
```

```
mmCategoryPerm: TMenuItem;
mmInfo: TMenuItem;
mmAbout: TMenuItem;
tbSaveDoc: TIBTable;
tbSaveDocDOC: TBlobField;
tbSaveDocID: TIntegerField;
tbSaveDocNAME: TIBStringField;
tbSaveDocFILENAME: TIBStringField;
tbSaveDocDESCRIPTION: TMemoField;
tSaveDoc: TIBTransaction;
sqlSect: TIBQuery;
sqlSectID: TIntegerField;
sqlSectNAME: TIBStringField;
tSect: TIBTransaction;
Panel1: TPanel;
leFind: TLabeledEdit;
Panel2: TPanel;
dbgDocRep: TDBGrid;
Panel3: TPanel;
btLoad: TButton;
btAdd: TButton;
btDel: TButton;
btEdit: TButton;
btUpdate: TButton;
Panel4: TPanel;
mDescription: TMemo;
mmDoc: TMenuItem;
mmDocType: TMenuItem;
mmDocArhiv: TMenuItem;
Panel7: TPanel;
btDocCategory: TButton;
leFindCategory: TLabeledEdit;
sqlCateg: TIBQuery;
tCateg: TIBTransaction;
dsCateg: TDataSource;
dbgCateg: TDBGrid;
sqlCategID: TIntegerField;
sqlCategNAME: TIBStringField;
sqlDocID: TIntegerField;
sqlDocNAME: TIBStringField;
sqlDocFILENAME: TIBStringField;
sqlDocDESCRIPTION: TMemoField;
sqlDocDOC: TBlobField;
sqlDocSTATE: TIntegerField;
sqlDocCATEGORY_ID: TIntegerField;
btRead: TButton;
mmLog: TMenuItem;
btDocArhiv: TButton;

procedure Fill();
procedure CategFill();
procedure DocFill();

procedure OpenDocument();
procedure CloseDocument();
procedure LookDocument();

procedure FormActivate(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure sqlDocCalcFields(DataSet: TDataSet);
procedure mmCloseClick(Sender: TObject);
procedure btAddClick(Sender: TObject);
procedure btEditClick(Sender: TObject);
procedure btDelClick(Sender: TObject);
procedure btLoadClick(Sender: TObject);
procedure btReadClick(Sender: TObject);
procedure leFindChange(Sender: TObject);
procedure dbgDocRepCellClick(Column: TColumn);
procedure mmClearUsersClick(Sender: TObject);
procedure mmAboutClick(Sender: TObject);
procedure btUpdateClick(Sender: TObject);
procedure mmUserListClick(Sender: TObject);
procedure btDocCategoryClick(Sender: TObject);
procedure chSectFiltrClick(Sender: TObject);
procedure mmOwnDataClick(Sender: TObject);
procedure mmActionsPermClick(Sender: TObject);
procedure mmCategoryPermClick(Sender: TObject);
```

```
    procedure mmDocTypeClick(Sender: TObject);
    procedure dbgCategCellClick(Column: TColumn);
    procedure dbgDocRepDblClick(Sender: TObject);
    procedure leFindCategoryChange(Sender: TObject);
    procedure mmDocArhivClick(Sender: TObject);
    procedure mmLogClick(Sender: TObject);
    procedure btDocArhivClick(Sender: TObject);

  private

  public
    WasActived: boolean;

    CurDoc: integer;
    CurCateg: integer;
  end;

var
  fmMain: TfmMain;

implementation

  uses Avtorizate, AddDoc, UserList, DocType, AddUser, ActionPerm, CategoryPerm, Log, Arhiv, About;

{$R *.dfm}

//---------------------------------------------------------------------------

procedure TfmMain.Fill();
begin
 CategFill(); //обновляем категории
 DocFill(); //обновляем список документов
end;

//---------------------------------------------------------------------------

  {заполнить список групп}
procedure TfmMain.CategFill();
var pos: integer;
    sql, where_p: string;
    keyword: string;
begin
 keyword:= AnsiUpperCase(leFindCategory.Text);
 if keyword <> '' then  where_p:= ' WHERE upper(NAME) LIKE '#39'%' + keyword + '%'#39;

 sql:= 'SELECT ID, NAME FROM DOCTYPE' + where_p + ' ORDER BY NAME';

 sqlCateg.SQL.Clear;
 sqlCateg.SQL.Add(sql);
 sqlCateg.Open;
 sqlCateg.FetchAll;

 if sqlCateg.RecordCount = 0 then
 begin
  CurCateg:= -1;
  exit;
 end;

 if not sqlCateg.Locate('Id', CurCateg, []) then  CurCateg:= sqlCategId.Value;
end;

//---------------------------------------------------------------------------

//заполнение таблицы документов
procedure TfmMain.DocFill();
var sql, where_p, inner_p: string;
    keyword: string;
begin

 //учёт прав доступа на документ у всех кроме админа
 inner_p:= '';
 if CurrentUserType <> 1 then
  inner_p:= 'INNER JOIN DocPerm DP ON (DP.USER_ID=' + IntToStr(CurrentUserID) + ' AND DP.DOCUMENT_ID=D.ID)';

 //поиск по ключевому слову
 keyword:= AnsiUpperCase(leFind.Text);

 where_p:= '';
```

```
 if keyword <> '' then where_p:= ' AND upper(D.NAME) LIKE '#39'%' + keyword + '%'#39;

 //собираем запрос
 sql:= 'SELECT D.* FROM Document D ' + inner_p + '  WHERE D.CATEGORY_ID=' + IntToStr(CurCateg) + where_p + ' ORDER BY
D.NAME';

 sqlDoc.SQL.Clear;
 sqlDoc.SQL.Add(sql);
 sqlDoc.Open;
 sqlDoc.FetchAll;

 if sqlDoc.RecordCount = 0 then
 begin
  CurDoc:= -1;

  btEdit.Enabled:= false;
  btDel.Enabled:= false;
  btLoad.Enabled:= false;
  btRead.Enabled:= false;
  btDocArhiv.Enabled:= false;

  mDescription.Clear;

  exit;
 end;

 if not sqlDoc.Locate('Id', CurDoc, []) then  CurDoc:= sqlDocId.Value;


 if fmS.DocStatus(sqlDocState.Value) <> 0 then btLoad.Caption:= 'Сохранить'
 else btLoad.Caption:= 'Редактировать';

 mDescription.Text:= sqlDocDescription.Value;

 btEdit.Enabled:= true;
 btDel.Enabled:= true;
 btLoad.Enabled:= true;
 btRead.Enabled:= true;

 btDocArhiv.Enabled:= fmS.IsArhiv(CurDoc);
end;


//---------------------------------------------------------------------------

procedure TfmMain.OpenDocument();
var DestDir, DestFN: string;
begin
 //проверка прав на просмотр документов
 if not fmS.IsActionPermition('doc_change') then
 begin
  MessageBox(handle, 'У вас нет прав на чтение документов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 //проверка прав на данный документ
 if not fmS.IsDocumentPermition(sqlDocID.Value) then
 begin
  MessageBox(handle, 'У вас нет прав на работу с данным документом','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;


 DestDir:= Dir + 'Doc';
 if not DirectoryExists(DestDir) then MkDir(DestDir);

 DestFN:= DestDir + '\' + sqlDocFileName.Value;
 if FileExists(DestFN) then
 begin
  MessageBox(handle, 'Выбранный файл уже находится в дирректории для редактирования. Возможно файл ранее был загружен и не
закрыт.','Ошибка!', MB_OK + MB_ICONEXCLAMATION);
  fmS.DelUserFromStatus(sqlDocId.Value);
  exit;
 end;

 sqlDocDoc.SaveToFile(DestFN);
 ShellExecute(application.handle, pchar('open'), pchar(DestFN), '', '', SW_SHOWNORMAL);
```

```
 fmS.AddUserInStatus(sqlDocId.Value, CurrentUserID);
end;

//---------------------------------------------------------------------------

procedure TfmMain.CloseDocument();
var DestFN: string;
    StateStr: string;
    success: boolean;
begin
 //проверка прав на просмотр документов
 if not fmS.IsActionPermition('doc_change') then
 begin
  MessageBox(handle, 'У вас нет прав на чтение документов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 //проверка прав на данный документ
 if not fmS.IsDocumentPermition(sqlDocID.Value) then
 begin
  MessageBox(handle, 'У вас нет прав на работу с данным документом','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 //проверка состояния самого документа
 if fmS.DocStatus(sqlDocState.Value) <> 1 then
  begin
  MessageBox(handle, 'Документ не является редактируемым. Возможно статус был сброшен администратором','Недостаточно прав',
MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;


 DestFN:= Dir + 'Doc\' + sqlDocFileName.Value;

 if not FileExists(DestFN) then
 begin
  MessageBox(handle, 'Открытый для редактирования файл отсутствует в дирректории. Загрузите его снова.','Ошибка!', MB_OK +
MB_ICONEXCLAMATION);
  fmS.DelUserFromStatus(sqlDocId.Value);
  exit;
 end;

 //создание архивной копии документа
 fmS.SaveToArhiv(sqlDocId.Value);

 success:= false;

 //сохранение изменённого документа
 if tSaveDoc.Active then tSaveDoc.Rollback;

 tSaveDoc.StartTransaction;
 try
  tbSaveDoc.Open;

  if not tbSaveDoc.Locate('Id', sqlDocId.Value, []) then
  begin
   tSaveDoc.Rollback;
   exit;
  end;

  tbSaveDoc.Edit;
  tbSaveDocDoc.LoadFromFile(DestFN);

  tbSaveDoc.Post;

  tSaveDoc.Commit;
  success:= true;
 except
  tSaveDoc.Rollback;
  tbSaveDoc.Cancel;
 raise;
 end;

 tbSaveDoc.Close;

 fmS.DelUserFromStatus(sqlDocId.Value);
 DeleteFile(DestFN);
```

```
 if success then
  fmS.SaveLog('Документ №' + IntToStr(sqlDocId.Value) + ' изменён');
end;

//---------------------------------------------------------------------------

//открытие документа для просмотра
procedure TfmMain.LookDocument();
var DestDir, DestFN: string;
begin
 //проверка прав на просмотр документов
 if not fmS.IsActionPermition('doc_read') then
 begin
  MessageBox(handle, 'У вас нет прав на чтение документов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 //проверка прав на данный документ
 if not fmS.IsDocumentPermition(sqlDocID.Value) then
 begin
  MessageBox(handle, 'У вас нет прав на просмотр данного документа','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;


 DestDir:= Dir + 'ReadDoc';
 if not DirectoryExists(DestDir) then MkDir(DestDir);

 DestFN:= DestDir + '\' + sqlDocFileName.Value;

 //если такой файл уже есть в директории для чтения - удаляем его
 if FileExists(DestFN) then
   DeleteFile(DestFN);

 sqlDocDOC.SaveToFile(DestFN);
 ShellExecute(application.handle, pchar('open'), pchar(DestFN), '', '', SW_SHOWNORMAL);
end;

//---------------------------------------------------------------------------

procedure TfmMain.FormActivate(Sender: TObject);
var Rez: integer;
begin
 if not WasActived then
 begin
  WasActived:= false;

   //подсоединение к базе данных
  if fmS.DB.Connected then fmS.DB.Connected:= false;

  fmS.DB.Params.Clear;
  fmS.DB.DatabaseName:= CFG.DBConfig.ServerName;
  fmS.DB.Params.Add(CFG.DBConfig.Username);
  fmS.DB.Params.Add(CFG.DBConfig.Password);
  fmS.DB.Params.Add('lc_ctype=WIN1251');

  fmS.DB.Connected:= true;

  //--Авторизация--//
  Rez:= fmAvtorizate.ShowModal;
  if Rez <> 5 then fmMain.Close
  else
   fmS.SaveLog('Авторизация в системе выполнена'); //лог
 end;

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.FormCreate(Sender: TObject);
begin
 WasActived:= false;
 Dir:= ExtractFilePath(Application.ExeName); //дирректория
 LoadCFG(); //загружаем настройки конфигурации

 CurCateg:= -1;
```

```
end;

//---------------------------------------------------------------------------

procedure TfmMain.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 if CurrentUserID <> 0 then
 begin
  fmS.DeactivateUser(CurrentUserID);
  fmS.SaveLog('Пользователь завершил работу'); //лог
 end;

 fmS.DB.Connected:= false;
 CurCateg:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmMain.sqlDocCalcFields(DataSet: TDataSet);
var RezStatus: integer;
begin
 RezStatus:= fmS.DocStatus(sqlDocState.Value);

 case RezStatus of
  0: sqlDockalk_state.Value:= 'закрыт';
  1: sqlDockalk_state.Value:= 'открыт';
  2: sqlDockalk_state.Value:= 'используется';
 end;
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmCloseClick(Sender: TObject);
begin
 Close;
end;

//---------------------------------------------------------------------------

procedure TfmMain.btAddClick(Sender: TObject);
begin

  //проверка прав на добавление документов
 if not fmS.IsActionPermition('doc_add') then
 begin
  MessageBox(handle, 'У вас нет прав на добавление документов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmAddDoc.RegState:= rsNew;
 fmAddDoc.CurSect:= CurCateg;
 fmAddDoc.ShowModal;

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.btEditClick(Sender: TObject);
begin

 //проверка прав на редактирование документов
 if not fmS.IsActionPermition('doc_edit') then
 begin
  MessageBox(handle, 'У вас нет прав на редактирование данных о документе','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 //вставить проверку на категорию !!!!

 fmAddDoc.RegState:= rsEdit;
 fmAddDoc.DocID:= CurDoc;
 fmAddDoc.ShowModal;

 Fill();
end;
```

```
//---------------------------------------------------------------------------

procedure TfmMain.btDelClick(Sender: TObject);
begin

 //проверка прав на редактирование документов
 if not fmS.IsActionPermition('doc_del') then
 begin
  MessageBox(handle, 'У вас нет прав на удаление документа','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 if CurDoc = -1 then exit;

 if sqlDocState.Value <> 0 then
 begin
  MessageBox(handle, 'Документ открыт для редактирования. Файл не может быть удалён.','Ошибка!', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 if IDCANCEL = MessageBox(Handle,'Уверены что хотите удалить выбранный документ?','Внимание...',MB_OKCANCEL +
MB_ICONEXCLAMATION) then exit;

 fmS.ExecuteSQL('DELETE FROM Document WHERE id=' + IntToStr(CurDoc));

 fmS.SaveLog('Удалён документ №' + IntToStr(CurDoc)); //лог

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.btLoadClick(Sender: TObject);
var RezStatus: integer;
begin
 if sqlDoc.RecordCount = 0 then exit;

 RezStatus:= fmS.DocStatus(sqlDocState.Value);

 case RezStatus of
  0: OpenDocument();
  1: CloseDocument();
  2: MessageBox(handle, 'В данный момент документ редактируется другим пользователем. Документ нельзя редактировать
одновременно','Ошибка!', MB_OK + MB_ICONEXCLAMATION);
 end;

 DocFill();
end;

//---------------------------------------------------------------------------

//открыть документ для просмотра
procedure TfmMain.btReadClick(Sender: TObject);
begin
 if sqlDoc.RecordCount = 0 then exit;

 LookDocument();
 DocFill();

 fmS.SaveLog('Документ №' + IntToStr(CurDoc) + ' открыт для просмотра'); //лог
end;

//---------------------------------------------------------------------------

procedure TfmMain.leFindChange(Sender: TObject);
begin
 DocFill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.dbgDocRepCellClick(Column: TColumn);
begin
 if sqlDoc.RecordCount = 0 then
 begin
  CurDoc:= -1;
  btLoad.Enabled:= false;
```

```
 btDocArhiv.Enabled:= false;
 exit;
end;

btLoad.Enabled:= true;

CurDoc:= sqlDocID.Value;
mDescription.Text:= sqlDocDescription.Value;

if fmS.DocStatus(sqlDocState.Value) = 1 then btLoad.Caption:= 'Закрыть'
else btLoad.Caption:= 'Открыть';

btDocArhiv.Enabled:= fmS.IsArhiv(CurDoc);
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmClearUsersClick(Sender: TObject);
begin

 //проверка прав на сброс активных пользователей
 if not fmS.IsActionPermition('user_clear') then
 begin
  MessageBox(handle, 'У вас нет прав на сброс активных пользователей','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmS.ClearActivUsers(CurrentUserId);
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmAboutClick(Sender: TObject);
begin
 fmAbout.ShowModal;
end;

//---------------------------------------------------------------------------

procedure TfmMain.btUpdateClick(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmUserListClick(Sender: TObject);
begin
 //проверка прав на управление пользователями
 if not fmS.IsActionPermition('user_clear') then
 begin
  MessageBox(handle, 'У вас нет прав на управление списком пользователей','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 fmUserList.ShowModal;
end;

//---------------------------------------------------------------------------

procedure TfmMain.btDocCategoryClick(Sender: TObject);
begin

  //проверка прав на редактирование категорий документов
 if not fmS.IsActionPermition('category_change') then
 begin
  MessageBox(handle, 'У вас нет прав на редактирование категорий документов','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 fmDocType.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------
```

```
procedure TfmMain.chSectFiltrClick(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmOwnDataClick(Sender: TObject);
begin
 fmAddUser.UserID:= CurrentUserID;
 fmAddUser.RegSt:= rsEdit;

 fmAddUser.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmActionsPermClick(Sender: TObject);
begin
   //проверка прав на распределение прав
 if not fmS.IsActionPermition('permittion_change') then
 begin
  MessageBox(handle, 'У вас нет доступа к системе распределения прав','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmActionPerm.ShowModal;
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmCategoryPermClick(Sender: TObject);
begin
    //проверка прав на распределение прав
 if not fmS.IsActionPermition('permittion_change') then
 begin
  MessageBox(handle, 'У вас нет прав на систему управления доступом к категориям документов','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 fmCategoryPerm.ShowModal;
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmDocTypeClick(Sender: TObject);
begin
  //проверка прав на редактирование категорий документов
 if not fmS.IsActionPermition('category_change') then
 begin
  MessageBox(handle, 'У вас нет прав на редактирование категорий документов','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 fmDocType.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.dbgCategCellClick(Column: TColumn);
begin
 if sqlCateg.RecordCount = 0 then
 begin
  CurCateg:= -1;
  exit;
 end;

 CurCateg:= sqlCategID.Value;
 DocFill(); //обновление списка документов
end;

//---------------------------------------------------------------------------

  //двойной клик - очистка статуса документа
```

```
procedure TfmMain.dbgDocRepDblClick(Sender: TObject);
begin
 if sqlDoc.RecordCount = 0 then exit;  //проверка на пустой список
 if not fmS.IsActionPermition('user_clear') then exit; //проверка прав на очистку статуса документа

 //проверка: используется ли документ
 if fmS.DocStatus(sqlDocState.Value) <> 2 then exit;

 //уведомление
 if IDCANCEL = MessageBox(Handle,'Вы обладаете правом на сброс состояния документа, редактируемого другим пользователем. Уверены что хотите сбросить состояние?','Внимание...',MB_OKCANCEL + MB_ICONEXCLAMATION) then exit;

 //сброс
 fmS.DelUserFromStatus(sqlDocId.Value);
 DocFill(); //обновление списка документов
end;

//---------------------------------------------------------------------------

procedure TfmMain.leFindCategoryChange(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmDocArhivClick(Sender: TObject);
begin
 fmArhiv.chDocFiltr.Checked:= false;
 fmArhiv.CurDoc:= -1;
 fmArhiv.CurArhiv:= -1;

 //проверка прав на восстановление архивных копий
 if not fmS.IsActionPermition('doc_recure') then
 begin
  MessageBox(handle, 'У вас нет прав на просмотр или восстановление копий  документов из архивов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmArhiv.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmMain.mmLogClick(Sender: TObject);
begin
   //проверка прав на просмотр лога
 if not fmS.IsActionPermition('log_read') then
 begin
  MessageBox(handle, 'У вас нет прав на просмотр лога работы пользователей с системой','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmLog.ShowModal;
end;

//---------------------------------------------------------------------------

procedure TfmMain.btDocArhivClick(Sender: TObject);
begin
 fmArhiv.chDocFiltr.Checked:= true;
 fmArhiv.CurDoc:= sqlDocId.Value;

 //проверка прав на восстановление архивных копий
 if not fmS.IsActionPermition('doc_recure') then
 begin
  MessageBox(handle, 'У вас нет прав на просмотр или восстановление копий  документов из архивов','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 fmArhiv.ShowModal;
 Fill();
end;
```

```
//---------------------------------------------------------------------------

end.
```

Модуль ActionPerm.pas

```
unit ActionPerm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, IBDatabase, IBCustomDataSet, IBQuery, StdCtrls,
  Grids, DBGrids, ExtCtrls, SQLDM, Global;

type
  TfmActionPerm = class(TForm)
    Panel1: TPanel;
    btAdd: TButton;
    btDel: TButton;
    sqlPermAction: TIBQuery;
    tPermAction: TIBTransaction;
    dsPermAction: TDataSource;
    dbgUser: TDBGrid;
    lbGroups: TLabel;
    cbGroups: TComboBox;
    sqlGroups: TIBQuery;
    sqlGroupsID: TIntegerField;
    sqlGroupsNAME: TIBStringField;
    tGroups: TIBTransaction;
    lbAction: TLabel;
    cbAction: TComboBox;
    sqlAction: TIBQuery;
    tAction: TIBTransaction;
    sqlPermActionID: TIntegerField;
    sqlPermActionGroupName: TIBStringField;
    sqlPermActionActionName: TIBStringField;
    sqlActionID: TIntegerField;
    sqlActionNAME: TIBStringField;
    sqlPermActionACTNAME: TIBStringField;
    sqlPermActionGROUP_ID: TIntegerField;

    procedure Fill();

    //группы
    procedure GroupsFill();
    function GroupsStandBy(id: integer): boolean;

    //действия
    procedure ActionFill();
    function ActionStandBy(id: integer): boolean;

    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btAddClick(Sender: TObject);
    procedure btDelClick(Sender: TObject);
    procedure dbgUserCellClick(Column: TColumn);
    procedure cbGroupsChange(Sender: TObject);
    procedure cbActionChange(Sender: TObject);

  private
    SvGroups: TDIntArr;
    SvAction: TDIntArr;
  public
    CurRule: integer;
    CurGroup: integer;
    CurAction: integer;
  end;

var
  fmActionPerm: TfmActionPerm;

implementation
```

```
{$R *.dfm}

//---------------------------------------------------------------------------

procedure TfmActionPerm.Fill();
begin
 //список групп
 GroupsFill();
 if Length(SvGroups) <> 0 then
 begin
  if CurGroup = -1 then
  begin
   cbGroups.ItemIndex:= 0;
   CurGroup:= SvGroups[0];
  end
  else if not GroupsStandBy(CurGroup) then
  begin
   cbGroups.ItemIndex:= 0;
   CurGroup:= SvGroups[0];
  end;
 end;


 //список действий
 ActionFill();
 if Length(SvAction) <> 0 then
 begin
  if CurAction = -1 then
  begin
   cbAction.ItemIndex:= 0;
   CurAction:= SvAction[0];
  end
  else if not ActionStandBy(CurAction) then
  begin
   cbAction.ItemIndex:= 0;
   CurAction:= SvAction[0];
  end;
 end;


 //список правил
 sqlPermAction.Close;
 sqlPermAction.Open;
 sqlPermAction.FetchAll;

 //интерфейс
 btAdd.Enabled:= false;
 btDel.Enabled:= false;

 if sqlPermAction.RecordCount = 0 then
  CurRule:= -1
 else
  btDel.Enabled:= true;

 if Length(SvAction) <> 0 then
  btAdd.Enabled:= true;

 if not sqlPermAction.Locate('id', CurRule, []) then CurRule:= sqlPermActionId.Value;
end;

//---------------------------------------------------------------------------

 //заполнить список групп
procedure TfmActionPerm.GroupsFill();
var pos: integer;
begin
 Finalize(SvGroups);
 cbGroups.Clear;

 sqlGroups.Close;
 sqlGroups.Open;
 sqlGroups.FetchAll;

 SetLength(SvGroups,sqlGroups.RecordCount);
 sqlGroups.First;
 pos:=0;
 while not sqlGroups.Eof do
 begin
```

```
  SvGroups[pos]:= sqlGroupsID.Value;
  cbGroups.Items.Add(sqlGroupsNAME.Value);
  inc(pos);
  sqlGroups.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать группу с индексом id. Если такой нет -1
function TfmActionPerm.GroupsStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvGroups) do
  if SvGroups[i] = id then
  begin
   Result:= true;
   cbGroups.ItemIndex:= i;
   exit;
  end;

 cbGroups.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

 //заполнить список действий
procedure TfmActionPerm.ActionFill();
var pos: integer;
    sql: string;
begin
 Finalize(SvAction);
 cbAction.Clear;

 if CurGroup = -1 then
 begin
   cbAction.Enabled:= false;
   CurAction:= -1;
   exit;
 end;

 cbAction.Enabled:= true;

 //запрос
 sql:= 'SELECT A.ID, A.NAME FROM ActionType A LEFT JOIN ActionPerm AP ON AP.ACTION_ID=A.ID AND AP.GROUP_ID=' +
IntToStr(CurGroup) + ' WHERE AP.ID IS NULL ORDER BY A.Name';

 sqlAction.Close;
 sqlAction.SQL.Clear;
 sqlAction.SQL.Add(sql);
 sqlAction.Open;
 sqlAction.FetchAll;

 SetLength(SvAction,sqlAction.RecordCount);
 sqlAction.First;
 pos:=0;
 while not sqlAction.Eof do
 begin
  SvAction[pos]:= sqlActionID.Value;
  cbAction.Items.Add(sqlActionNAME.Value);
  inc(pos);
  sqlAction.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать действие с индексом id. Если такого нет -1
function TfmActionPerm.ActionStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvAction) do
  if SvAction[i] = id then
  begin
   Result:= true;
   cbAction.ItemIndex:= i;
```

```
   exit;
  end;

 cbAction.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.FormCreate(Sender: TObject);
begin
 CurRule:= -1;
 CurGroup:= -1;
 CurAction:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.FormActivate(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurRule:= -1;
 CurGroup:= -1;
 CurAction:= -1;

 sqlPermAction.Close;
 sqlGroups.Close;
 sqlAction.Close;
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.btAddClick(Sender: TObject);
begin
 if CurGroup = -1 then exit;
 if CurAction = -1 then exit;

 fmS.ExecuteSQL('INSERT INTO ActionPerm (Group_id, Action_id) VALUES (' + IntToStr(CurGroup) + ',' + IntToStr(CurAction) + ')');

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.btDelClick(Sender: TObject);
begin
 if CurRule = -1 then exit;

 if (sqlPermActionACTNAME.Value = 'permittion_change') and (sqlPermActionGROUP_ID.Value = 1) then
 begin
  MessageBox(handle, 'Вы не можете удалить у администратора право на редактирование разрешений для учётных запиисей','Ошибка!',
MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 if IDCANCEL = MessageBox(Handle,'Уверены что хотите удалить выбранное правило?','Внимание...',MB_OKCANCEL +
MB_ICONEXCLAMATION) then exit;

 fmS.ExecuteSQL('DELETE FROM ActionPerm WHERE id=' + IntToStr(CurRule));

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.dbgUserCellClick(Column: TColumn);
begin
 if sqlPermAction.RecordCount = 0 then exit;
 CurRule:= sqlPermActionId.Value;
end;

//---------------------------------------------------------------------------
```

```
procedure TfmActionPerm.cbGroupsChange(Sender: TObject);
begin
 if cbGroups.ItemIndex = -1  then exit;
 CurGroup:= SvGroups[cbGroups.ItemIndex];
 sqlGroups.Locate('id',CurGroup,[]);

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmActionPerm.cbActionChange(Sender: TObject);
begin
 if cbAction.ItemIndex = -1  then exit;
 CurAction:= SvAction[cbAction.ItemIndex];
 sqlAction.Locate('id',CurAction,[]);
end;

//---------------------------------------------------------------------------

end.
```

Модуль AddDoc.pas

```
unit AddDoc;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, SQLDM, Global, DB, IBCustomDataSet, IBTable,
  IBDatabase, IBQuery;

type
  TfmAddDoc = class(TForm)
    mDescription: TMemo;
    Label1: TLabel;
    leName: TLabeledEdit;
    btOK: TButton;
    btCancel: TButton;
    Label2: TLabel;
    lbFileName: TLabel;
    btOpenFile: TButton;
    odOpen: TOpenDialog;
    tbDoc: TIBTable;
    tDoc: TIBTransaction;
    Label3: TLabel;
    cbSect: TComboBox;
    btDocType: TButton;
    sqlSect: TIBQuery;
    sqlSectID: TIntegerField;
    sqlSectNAME: TIBStringField;
    tSect: TIBTransaction;
    tbDocID: TIntegerField;
    tbDocNAME: TIBStringField;
    tbDocFILENAME: TIBStringField;
    tbDocDESCRIPTION: TMemoField;
    tbDocDOC: TBlobField;
    tbDocSTATE: TIntegerField;
    tbDocCATEGORY_ID: TIntegerField;

    procedure ClearFields();
    function LoadData(): boolean;
    function Check():boolean;

    procedure SectFill();
    function SectStandBy(id: integer): boolean;

    procedure btOKClick(Sender: TObject);
    procedure btCancelClick(Sender: TObject);
    procedure btOpenFileClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btDocTypeClick(Sender: TObject);
```

```
    procedure cbSectChange(Sender: TObject);
  private
    CurFileName: string;
    OpenError: boolean;
    SvSect: TDIntArr;
  public
    RegState: TRegStatus;
    DocID: integer;
    CurSect: integer;
  end;

var
  fmAddDoc: TfmAddDoc;

implementation

{$R *.dfm}

  uses DocType;

procedure TfmAddDoc.ClearFields();
begin
 leName.Text:= '';
 lbFileName.Caption:= '';
 mDescription.Clear;
end;

//---------------------------------------------------------------------------

function TfmAddDoc.LoadData(): boolean;
begin
 Result:= false;
 fmS.OpenSQL('SELECT * FROM Document WHERE id=' + IntToStr(DocID));
 if fmS.sqlData.RecordCount = 0 then exit;

 CurFileName:= fmS.sqlData.FieldByName('FileName').AsString;
 lbFileName.Caption:= fmS.sqlData.FieldByName('FileName').AsString;
 leName.Text:= fmS.sqlData.FieldByName('Name').AsString;
 mDescription.Text:= fmS.sqlData.FieldByName('Description').AsString;
 CurSect:= fmS.sqlData.FieldByName('Category_ID').AsInteger;

 Result:= true;
end;

//---------------------------------------------------------------------------

function TfmAddDoc.Check():boolean;
begin
 Result:= false;

 if leName.Text = '' then
 begin
  MessageBox(handle, 'Введите название документа', 'Ошибка ввода', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 if (RegState = rsNew) and not FileExists(CurFileName) then
 begin
  MessageBox(handle, 'Файл не найден. Выберите файл документа', 'Ошибка ввода', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 if CurSect = -1 then
 begin
  MessageBox(handle, 'Группа к которой относится документ не выбрана', 'Ошибка ввода', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;


 Result:= true;
end;

//---------------------------------------------------------------------------

  {заполнить список групп}
procedure TfmAddDoc.SectFill();
var pos: integer;
    sql: string;
```

```
begin
 Finalize(SvSect);
 cbSect.Clear;

 sqlSect.Close;
 sqlSect.Open;
 sqlSect.FetchAll;

 SetLength(SvSect,sqlSect.RecordCount);
 sqlSect.First;
 pos:=0;
 while not sqlSect.Eof do
 begin
  SvSect[pos]:= sqlSectId.Value;
  cbSect.Items.Add(sqlSectName.Value);
  inc(pos);
  sqlSect.Next;
 end;

 sqlSect.Close;
end;

//---------------------------------------------------------------------------

  {выбрать группу с индексом id. Если такой нет -1}
function TfmAddDoc.SectStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvSect) do
  if SvSect[i] = id then
  begin
   Result:= true;
   cbSect.ItemIndex:= i;
   exit;
  end;

 cbSect.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.btOKClick(Sender: TObject);
var success: boolean;
    NewID: integer;
begin
 if OpenError then exit;
 if not Check() then exit;

 success:= false;

 if tDoc.Active then tDoc.Rollback;

 tDoc.StartTransaction;
 try
  tbDoc.Open;

  if RegState = rsNew then
  begin
   tbDoc.Insert;
   tbDocDoc.LoadFromFile(CurFileName);
   tbDocSTATE.Value:= 0;
  end
  else
  begin

   if not tbDoc.Locate('Id', DocID, [])  then
   begin
    tDoc.Rollback;
    tbDoc.Close;
    exit;
   end;

   tbDoc.Edit;
  end;

  tbDocName.Value:= leName.Text;
  tbDocFileName.Value:= ExtractFileName(CurFileName);
```

```
 tbDocDescription.Value:= mDescription.Text;
 tbDocCategory_ID.Value:= CurSect;

 tbDoc.Post;
 tDoc.Commit;
 success:= true;
except
 tDoc.Rollback;
 tbDoc.Cancel;
raise;
end;


if success then
begin
 if RegState = rsNew then
 begin
  NewID:= fmS.GetLastDocumentByName(leName.Text);

  //лог о создании
  fmS.SaveLog('Создан документ №' + IntToStr(NewID) + ' ' + leName.Text);

  //доступ на документ создателю
  if CurrentUserType <> 1 then
  begin
   fmS.AddDocumentPermition(NewID, CurrentUserID);
   fmS.SaveLog('Предоставлен доступ пользователю №' + IntToStr(CurrentUserID) + ' к документу №' + IntToStr(NewID)); //лог
  end
 end
 else
  fmS.SaveLog('Внесены изменения в описание документа №' + IntToStr(DocID)); //лог об изменении
end;

tbDoc.Close;

 Close;
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.btCancelClick(Sender: TObject);
begin
 Close;
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.btOpenFileClick(Sender: TObject);
begin
 if RegState <> rsNew then exit;
 if not odOpen.Execute then exit;

 CurFileName:= odOpen.FileName;
 lbFileName.Caption:= ExtractFileName(odOpen.FileName);
 leName.Text:= lbFileName.Caption;
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.FormActivate(Sender: TObject);
begin
 OpenError:= false;

 if RegState = rsNone then OpenError:= true;
 if (RegState = rsEdit) and (DocID = -1) then OpenError:= true;

 ClearFields();
 if RegState = rsEdit then
 begin
  if not LoadData() then OpenError:= true;
  btOpenFile.Enabled:= false;
  fmAddDoc.Caption:= 'Изменение данных о документе';
 end
 else
 begin
  btOpenFile.Enabled:= true;
  fmAddDoc.Caption:= 'Добавление документа';
 end;
```

```
 if OpenError then
 begin
  MessageBox(handle, 'Произошла ошибка. Закройте это окно и обратитесь к администратору базы', 'Ошибка!', MB_OK +
MB_ICONEXCLAMATION);
  Close;
 end;

  //обновляем группы
 SectFill();
 if CurSect = -1 then cbSect.ItemIndex:= -1
 else
  if not SectStandBy(CurSect) then CurSect:= -1; //группа не найдена
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.FormCreate(Sender: TObject);
begin
 CurFileName:= '';
 DocID:= -1;
 RegState:= rsNone;
 CurSect:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurFileName:= '';
 DocID:= -1;
 RegState:= rsNone;
 CurSect:= -1;

 tbDoc.Close;
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.btDocTypeClick(Sender: TObject);
begin

 //проверка прав на редактирование категорий документов
 if not fmS.IsActionPermition('category_change') then
 begin
  MessageBox(handle, 'У вас нет прав на редактирование категорий документов','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
   exit;
 end;

 fmDocType.ShowModal;

 //обновляем группы
 SectFill();
 if CurSect = -1 then cbSect.ItemIndex:= -1
 else
  if not SectStandBy(CurSect) then CurSect:= -1; //группа не найдена
end;

//---------------------------------------------------------------------------

procedure TfmAddDoc.cbSectChange(Sender: TObject);
begin
 if cbSect.ItemIndex = -1 then exit;
 CurSect:= SvSect[cbSect.ItemIndex];
end;

//---------------------------------------------------------------------------

end.
```

Модуль AddUser.pas

```
unit AddUser;
```

```
interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, SQLDM, Global, IBDatabase, DB,
  IBCustomDataSet, IBQuery;

type
  TfmAddUser = class(TForm)
    leFio: TLabeledEdit;
    leLogin: TLabeledEdit;
    lePassword: TLabeledEdit;
    leConfirm: TLabeledEdit;
    btOK: TButton;
    btCancel: TButton;
    cbGroups: TComboBox;
    lbUserLevel: TLabel;
    sqlGroups: TIBQuery;
    tGroups: TIBTransaction;
    sqlGroupsID: TIntegerField;
    sqlGroupsNAME: TIBStringField;

    procedure ClearFields();
    function LoadData(): boolean;

    function Check(): boolean;

    //группы
    procedure GroupsFill();
    function GroupsStandBy(id: integer): boolean;


    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btOKClick(Sender: TObject);
    procedure btCancelClick(Sender: TObject);
    procedure cbGroupsChange(Sender: TObject);
  private
    SvGroups: TDIntArr;
  public
    UserID: integer;
    CurGroup: integer;
    SaveGroupID: integer; //сохранение ничального уровня прав при редактировании
    RegSt: TRegStatus;
  end;

var
  fmAddUser: TfmAddUser;

implementation

{$R *.dfm}

  //заполнить список групп
procedure TfmAddUser.GroupsFill();
var pos: integer;
begin
 Finalize(SvGroups);
 cbGroups.Clear;

 sqlGroups.Close;
 sqlGroups.Open;
 sqlGroups.FetchAll;

 SetLength(SvGroups,sqlGroups.RecordCount);
 sqlGroups.First;
 pos:=0;
 while not sqlGroups.Eof do
 begin
  SvGroups[pos]:= sqlGroupsID.Value;
  cbGroups.Items.Add(sqlGroupsNAME.Value);
  inc(pos);
  sqlGroups.Next;
 end;
end;

//---------------------------------------------------------------------------
```

```
  //выбрать группу с индексом id. Если такой нет -1
function TfmAddUser.GroupsStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvGroups) do
  if SvGroups[i] = id then
  begin
   Result:= true;
   cbGroups.ItemIndex:= i;
   exit;
  end;

 cbGroups.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmAddUser.ClearFields();
begin
 leLogin.Text:= '';
 leFio.Text:= '';
 lePassword.Text:= '';
 leConfirm.Text:= '';
 cbGroups.ItemIndex:= 0;
 cbGroups.Visible:= false;
 lbUserLevel.Visible:= false;
end;

//---------------------------------------------------------------------------

function TfmAddUser.LoadData(): boolean;
begin
 Result:= false;

 fmS.OpenSQL('SELECT * FROM DBUSERS WHERE id=' + IntToStr(UserID));
 if fmS.sqlData.RecordCount = 0 then
 begin
  MessageBox(handle, 'Пользователь не найден. Закройте это окно и обратитесь к администратору','Ошибка!', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 leLogin.Text:= fmS.sqlData.FieldByName('U_LOGIN').AsString;
 leFio.Text:= fmS.sqlData.FieldByName('FIO').AsString;
 lePassword.Text:=  fmS.sqlData.FieldByName('U_PASWORD').AsString;
 leConfirm.Text:=  fmS.sqlData.FieldByName('U_PASWORD').AsString;
 SaveGroupID:= fmS.sqlData.FieldByName('GROUP_ID').AsInteger;
 CurGroup:= SaveGroupID;

 fmAddUser.Caption:= fmS.sqlData.FieldByName('FIO').AsString + ' - изменение данных';

 Result:= true;
end;

//---------------------------------------------------------------------------

function TfmAddUser.Check(): boolean;
var sql: string;
begin
 Result:= false;

 //логин
 if leLogin.Text = '' then
 begin
  MessageBox(handle, 'Имя пользователя (Логин) не может быть пустым','Неверный ввод!', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 //уникальность логина
 sql:= 'SELECT id FROM DBUSERS WHERE upper(u_login)='#39 + AnsiUpperCase(leLogin.Text) + #39;
 if RegSt = rsEdit then sql:= sql + ' AND id<>' + IntToStr(UserID);
 fmS.OpenSQL(sql);
 if fmS.sqlData.RecordCount <> 0 then
 begin
  MessageBox(handle, 'В базе уже числится пользователь с таким Логином. Введите другой.','Неверный ввод!', MB_OK +
MB_ICONEXCLAMATION);
```

```
 exit;
end;

//пароль
if lePassword.Text = '' then
begin
 MessageBox(handle, 'Пароль не введён','Неверный ввод!', MB_OK + MB_ICONEXCLAMATION);
 exit;
end;

//соответствие пароля и подтверждения
if lePassword.Text <> leConfirm.Text then
begin
 MessageBox(handle, 'Пароль не совпадает с подтверждением пароля','Неверный ввод!', MB_OK + MB_ICONEXCLAMATION);
 exit;
end;

//фио
if leFIO.Text = '' then
begin
 MessageBox(handle, 'Введите ФИО пользователя','Неверный ввод!', MB_OK + MB_ICONEXCLAMATION);
 exit;
end;


//смена статуса при редактировании
if (RegSt = rsEdit) and (CurGroup <> SaveGroupID) then
begin

 //админ не может менять права своей записи
 if UserID = CurrentUserID then
 begin
  MessageBox(handle, 'Вы не можете менять уровень прав своей собственной учётной записи','Неверный ввод!', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 //админ на может менять права активному пользователю
 if fmS.IsActivUser(UserID) then
 begin
  MessageBox(handle, 'Данный пользователь сейчас работает в базе. Нельзя изменить уровень прав активного пользователя','Неверный
ввод!', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

end;

 Result:= true;
end;

//---------------------------------------------------------------------------

procedure TfmAddUser.FormCreate(Sender: TObject);
begin
 UserID:= -1;
 RegSt:= rsNone;
end;

//---------------------------------------------------------------------------

procedure TfmAddUser.FormActivate(Sender: TObject);
var FindError: boolean;
begin
 btOk.Enabled:= false;  //блокирует кнопку

 if (not fmS.IsActionPermition('user_change')) and (CurrentUserId <> UserID) then
 begin
  MessageBox(handle, 'У вас нет прав редактировать данные чужих учётных записей','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
  Close();
  exit;
 end;

 FindError:= true;    //статус ошибки
 if RegSt = rsNone then FindError:= false
 else
  if (RegSt = rsEdit) and (UserID = -1) then FindError:= false;
```

```
 if not FindError then
 begin
  MessageBox(handle, 'Произошла ошибка. Закройте это окно и обратитесь к администратору базы','Ошибка', MB_OK +
MB_ICONEXCLAMATION);
  Close();
  exit;
 end;

 ClearFields();
 if RegSt = rsNew then fmAddUser.Caption:= 'Добавление пользователя'
 else FindError:= LoadData();

 if FindError then btOk.Enabled:= true;  //снимаем блокировку

 //список групп
 GroupsFill();
 if Length(SvGroups) <> 0 then
 begin
  if CurGroup = -1 then
  begin
   cbGroups.ItemIndex:= 0;
   CurGroup:= SvGroups[0];
  end
  else if not GroupsStandBy(CurGroup) then
  begin
   cbGroups.ItemIndex:= 0;
   CurGroup:= SvGroups[0];
  end;
 end;

 //для пользователей прячем панель выбора прав
 if fmS.IsActionPermition('user_change') then
 begin
  cbGroups.Visible:= true;
  lbUserLevel.Visible:= true;
 end;

end;

//---------------------------------------------------------------------------

procedure TfmAddUser.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 UserID:= -1;
 CurGroup:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmAddUser.btOKClick(Sender: TObject);
var sql: string;
    fio, login, userpassword, group_id: string;
begin
 if not sqlGroups.Locate('id',CurGroup,[]) then exit;
 if not Check() then exit;

 //получаем данные
 fio:= leFio.Text;
 login:= leLogin.Text;
 userpassword:= lePassword.Text;
 group_id:= IntToStr(CurGroup);

 //готовим запрос
 case RegSt of
  rsNew: sql:= 'INSERT INTO DBUSERS (FIO, U_LOGIN, U_PASWORD, STATE, GROUP_ID) VALUES ('#39 + fio + #39', '#39 + login +
#39', '#39 + userpassword + #39', 1, '#39 + group_id + #39')';
  rsEdit: sql:= 'UPDATE DBUSERS SET FIO='#39 + fio + #39', U_LOGIN='#39 + login + #39', U_PASWORD='#39 + userpassword + #39',
STATE=1, GROUP_ID=' + group_id + ' WHERE id=' + IntToStr(UserID);
 end;

// InputQuery('','',sql);

 //выполняем запрос
 fmS.ExecuteSQL(sql);

 Close();
end;
```

```
//---------------------------------------------------------------------------

procedure TfmAddUser.btCancelClick(Sender: TObject);
begin
 Close();
end;

//---------------------------------------------------------------------------

procedure TfmAddUser.cbGroupsChange(Sender: TObject);
begin
 if cbGroups.ItemIndex = -1  then exit;
 CurGroup:= SvGroups[cbGroups.ItemIndex];
 sqlGroups.Locate('id',CurGroup,[]);
end;

//---------------------------------------------------------------------------

end.
```

Модуль Arhiv.pas

```
unit Arhiv;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, IBDatabase, IBCustomDataSet, IBQuery, StdCtrls,
  Grids, DBGrids, ExtCtrls, SQLDM, Global;

type
  TfmArhiv = class(TForm)
    Panel1: TPanel;
    btRecure: TButton;
    btClose: TButton;
    sqlArhiv: TIBQuery;
    tArhiv: TIBTransaction;
    dsArhiv: TDataSource;
    dbgArhiv: TDBGrid;
    lbAction: TLabel;
    cbDoc: TComboBox;
    sqlDoc: TIBQuery;
    tDoc: TIBTransaction;
    sqlDocID: TIntegerField;
    sqlDocCatName: TIBStringField;
    sqlDocDocName: TIBStringField;
    chDocFiltr: TCheckBox;
    sqlArhivID: TIntegerField;
    sqlArhivSAVE_DATE: TDateTimeField;
    sqlArhivCatName: TIBStringField;
    sqlArhivDocName: TIBStringField;
    sqlArhivDocNum: TIntegerField;

    procedure FillArhiv();

    //документы
    procedure DocFill();
    function DocStandBy(id: integer): boolean;

    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btRecureClick(Sender: TObject);
    procedure btCloseClick(Sender: TObject);
    procedure dbgArhivCellClick(Column: TColumn);
    procedure cbDocChange(Sender: TObject);
    procedure chDocFiltrClick(Sender: TObject);

  private
    SvDoc: TDIntArr;
  public
    CurArhiv: integer;
    CurDoc: integer;
  end;
```

```
var
  fmArhiv: TfmArhiv;

implementation

{$R *.dfm}

//---------------------------------------------------------------------------

procedure TfmArhiv.FillArhiv();
var sql, where_p: string;
begin

 //фильтр по документу
 where_p:= '';
 if chDocFiltr.Checked then
  where_p:= ' WHERE D.ID=' + IntToStr(CurDoc);

  //запрос
 sql:= 'SELECT A.ID, A.SAVE_DATE, DT.NAME as "CatName", D.NAME as "DocName", D.ID as "DocNum" FROM DOCUMENTARHIV
A INNER JOIN DOCUMENT D ON D.ID=A.DOCUMENT_ID INNER JOIN DOCTYPE DT ON DT.ID=D.CATEGORY_ID ' + where_p + '
ORDER BY D.Name, D.ID, A.SAVE_DATE DESC';

 sqlArhiv.Close;
 sqlArhiv.SQL.Clear;
 sqlArhiv.SQL.Add(sql);
 sqlArhiv.Open;
 sqlArhiv.FetchAll;

 //интерфейс
 btRecure.Enabled:= false;

 if sqlArhiv.RecordCount = 0 then
  CurArhiv:= -1
 else
  btRecure.Enabled:= true;

 if not sqlArhiv.Locate('id', CurArhiv, []) then CurArhiv:= sqlArhivId.Value;
end;

//---------------------------------------------------------------------------

 //заполнить список документов
procedure TfmArhiv.DocFill();
var pos: integer;
    sql: string;
begin
 Finalize(SvDoc);
 cbDoc.Clear;

 sqlDoc.Close;
 sqlDoc.Open;
 sqlDoc.FetchAll;

 SetLength(SvDoc,sqlDoc.RecordCount);
 sqlDoc.First;
 pos:=0;
 while not sqlDoc.Eof do
 begin
  SvDoc[pos]:= sqlDocID.Value;
  cbDoc.Items.Add(sqlDocCatName.Value + ' \ №' + sqlDocId.AsString + ' ' + sqlDocDocName.Value);
  inc(pos);
  sqlDoc.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать документ с индексом id. Если такого нет -1
function TfmArhiv.DocStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvDoc) do
  if SvDoc[i] = id then
  begin
   Result:= true;
   cbDoc.ItemIndex:= i;
```

```
   exit;
  end;

 cbDoc.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.FormCreate(Sender: TObject);
begin
 CurArhiv:= -1;
 CurDoc:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.FormActivate(Sender: TObject);
begin
  //список документов
 DocFill();
 if Length(SvDoc) <> 0 then
 begin
  if CurDoc = -1 then
  begin
   cbDoc.ItemIndex:= 0;
   CurDoc:= SvDoc[0];
  end
  else if not DocStandBy(CurDoc) then
  begin
   cbDoc.ItemIndex:= 0;
   CurDoc:= SvDoc[0];
  end;
 end;

 FillArhiv();
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurArhiv:= -1;
 CurDoc:= -1;

 sqlArhiv.Close;
 sqlDoc.Close;
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.btRecureClick(Sender: TObject);
begin
 if CurArhiv = -1 then exit;
 if CurDoc = -1 then exit;

 //проверка прав
 if not fmS.IsActionPermition('doc_recure') then
 begin
   MessageBox(handle, 'У вас нет прав на восстановление документов из архивных копий','Недостаточно прав', MB_OK +
MB_ICONEXCLAMATION);
   exit;
 end;

 //проверка доступа к документу
 if not fmS.IsDocumentPermition(CurDoc) then
 begin
   MessageBox(handle, 'У вас нет прав на работу с данным документом','Недостаточно прав', MB_OK + MB_ICONEXCLAMATION);
   exit;
 end;

 //подтверждение
 if IDCANCEL = MessageBox(Handle,'Уверены что хотите восстановить данную версию документа? Текущая версия будет
сохранена.','Внимание...',MB_OKCANCEL + MB_ICONEXCLAMATION) then exit;

 //восстановление
 fmS.LoadFromArhiv(CurArhiv, CurDoc);

 //уведомление
```

```
 MessageBox(handle, 'Документ восстановлен из архивной копии','Выполнено', MB_OK);

 FillArhiv();
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.btCloseClick(Sender: TObject);
begin
 Close();
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.dbgArhivCellClick(Column: TColumn);
begin
 if sqlArhiv.RecordCount = 0 then exit;
 CurArhiv:= sqlArhivId.Value;
 CurDoc:= sqlArhivDocNum.Value;
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.cbDocChange(Sender: TObject);
begin
 if cbDoc.ItemIndex = -1  then exit;
 CurDoc:= SvDoc[cbDoc.ItemIndex];
 sqlDoc.Locate('id',CurDoc,[]);

 FillArhiv();
end;

//---------------------------------------------------------------------------

procedure TfmArhiv.chDocFiltrClick(Sender: TObject);
begin
 FillArhiv();
end;

//---------------------------------------------------------------------------

end.
```

Модуль Avtorizate.pas

```
unit Avtorizate;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DBCtrls, StdCtrls, ExtCtrls, Global, DB, DBTables, SQLDM,
  IBCustomDataSet, IBQuery, IBDatabase;

type
  TfmAvtorizate = class(TForm)
    lePasword: TLabeledEdit;
    btOK: TButton;
    btCancel: TButton;
    Label1: TLabel;
    cbUser: TComboBox;
    sqlUser: TIBQuery;
    tUser: TIBTransaction;
    sqlUserID: TIntegerField;
    sqlUserFIO: TIBStringField;
    sqlUserU_LOGIN: TIBStringField;
    sqlUserU_PASWORD: TIBStringField;
    sqlUserSTATE: TIntegerField;
    sqlUserGROUP_ID: TIntegerField;

    procedure UserFill();
    function UserStandBy(id: integer): boolean;

      {Закрытие формы}
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
      {активизация формы}
    procedure FormActivate(Sender: TObject);
      {Кнопка <Войти>}
    procedure btOKClick(Sender: TObject);
      {Кнопка <Отмена>}
    procedure btCancelClick(Sender: TObject);
    procedure cbUserChange(Sender: TObject);
  private
    Rezultat: integer;
    SvUser: TDIntArr;
  public
    CurUser: integer;
  end;

var
  fmAvtorizate: TfmAvtorizate;

implementation

{$R *.dfm}

//---------------------------------------------------------------------------

  {заполнить список налоговых отчётностей}
procedure TfmAvtorizate.UserFill();
var pos: integer;
begin
 Finalize(SvUser);
 cbUser.Clear;

 sqlUser.Close;
 sqlUser.Open;
 sqlUser.FetchAll;

 SetLength(SvUser,sqlUser.RecordCount);
 sqlUser.First;
 pos:=0;
 while not sqlUser.Eof do
 begin
  SvUser[pos]:= sqlUserID.Value;
  cbUser.Items.Add(sqlUserU_LOGIN.Value + ' - ' + sqlUserFio.Value);
  inc(pos);
  sqlUser.Next;
 end;
end;

//---------------------------------------------------------------------------

  {выбрать должность с индексом id. Если такой нет -1}
function TfmAvtorizate.UserStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvUser) do
  if SvUser[i] = id then
  begin
   Result:= true;
   cbUser.ItemIndex:= i;
   exit;
  end;

 cbUser.ItemIndex:= -1; //запись не найдена
end;

//===========================================================================

  {Закрытие формы}
procedure TfmAvtorizate.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
 sqlUser.Close;
 ModalResult:= Rezultat;
end;

//---------------------------------------------------------------------------

  {активизация формы}
```

```
procedure TfmAvtorizate.FormActivate(Sender: TObject);
begin
 Rezultat:= -1;

  //список пользователей
 UserFill();
 if Length(SvUser) <> 0 then
 begin
  if CurUser = -1 then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end
  else if not UserStandBy(CurUser) then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end;
 end;

end;

//---------------------------------------------------------------------------

  {Кнопка <Войти>}
procedure TfmAvtorizate.btOKClick(Sender: TObject);
begin
 if not sqlUser.Locate('id',CurUser,[]) then exit;

 //пароль
 if (lePasword.Text <> sqlUserU_PASWORD.Value) then
 begin
  MessageBox(handle,'Пароль не верен','Ошибка авторизации!',MB_OK + MB_ICONEXCLAMATION);
  lePasword.Text:= '';
  exit;
 end;

 //активные пользователи
 if not fmS.ActivateUser(CurUser) then
 begin
  MessageBox(handle,'Данный пользователь уже работает','Ошибка доступа!',MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 CurrentUserID:= CurUser;
 CurrentUserType:= sqlUserGROUP_ID.Value;

 Rezultat:= 5;
 Close;
end;

//---------------------------------------------------------------------------

  {Кнопка <Отмена>}
procedure TfmAvtorizate.btCancelClick(Sender: TObject);
begin
 if IDYES <> MessageBox(Handle,'Отказ от авторизации приведёт к выходу из программы. Уверены что хотите
выйти?','Внимание...',MB_YESNO + MB_ICONEXCLAMATION) then exit;
 Close;
end;

//---------------------------------------------------------------------------

procedure TfmAvtorizate.cbUserChange(Sender: TObject);
begin
 if cbUser.ItemIndex = -1  then exit;
 CurUser:= SvUser[cbUser.ItemIndex];
 sqlUser.Locate('id',CurUser,[]);
end;

//---------------------------------------------------------------------------

end.
```

Модуль CategoryPerm.pas

```
unit CategoryPerm;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, IBDatabase, IBCustomDataSet, IBQuery, StdCtrls,
  Grids, DBGrids, ExtCtrls, SQLDM, Global;

type
  TfmCategoryPerm = class(TForm)
    Panel1: TPanel;
    btAdd: TButton;
    btDel: TButton;
    sqlPermDoc: TIBQuery;
    tPermDoc: TIBTransaction;
    dsPermDoc: TDataSource;
    dbgPermDoc: TDBGrid;
    lbGroups: TLabel;
    cbUser: TComboBox;
    sqlUser: TIBQuery;
    tUser: TIBTransaction;
    lbAction: TLabel;
    cbDoc: TComboBox;
    sqlDoc: TIBQuery;
    tDoc: TIBTransaction;
    sqlUserID: TIntegerField;
    sqlUserFIO: TIBStringField;
    sqlDocID: TIntegerField;
    sqlDocCatName: TIBStringField;
    sqlDocDocName: TIBStringField;
    sqlPermDocID: TIntegerField;
    sqlPermDocCatName: TIBStringField;
    sqlPermDocDocName: TIBStringField;

    procedure Fill();

    //пользователи
    procedure UserFill();
    function UserStandBy(id: integer): boolean;

    //документы
    procedure DocFill();
    function DocStandBy(id: integer): boolean;

    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btAddClick(Sender: TObject);
    procedure btDelClick(Sender: TObject);
    procedure dbgPermDocCellClick(Column: TColumn);
    procedure cbUserChange(Sender: TObject);
    procedure cbDocChange(Sender: TObject);

  private
    SvUser: TDIntArr;
    SvDoc: TDIntArr;
  public
    CurRule: integer;
    CurUser: integer;
    CurDoc: integer;
  end;

var
  fmCategoryPerm: TfmCategoryPerm;

implementation

{$R *.dfm}

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.Fill();
var sql: string;
begin
 //список пользователей
 UserFill();
 if Length(SvUser) <> 0 then
```

```
 begin
  if CurUser = -1 then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end
  else if not UserStandBy(CurUser) then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end;
 end;


 //список документов
 DocFill();
 if Length(SvDoc) <> 0 then
 begin
  if CurDoc = -1 then
  begin
   cbDoc.ItemIndex:= 0;
   CurDoc:= SvDoc[0];
  end
  else if not DocStandBy(CurDoc) then
  begin
   cbDoc.ItemIndex:= 0;
   CurDoc:= SvDoc[0];
  end;
 end;


 //список правил

  //запрос
 sql:= 'SELECT CP.ID, DT.NAME as "CatName", D.NAME as "DocName" FROM DocPerm CP INNER JOIN DBUsers U ON U.ID=CP.USER_ID INNER JOIN Document D ON D.ID=CP.DOCUMENT_ID INNER JOIN DocType DT ON DT.ID=D.CATEGORY_ID WHERE CP.USER_ID=' + IntToStr(CurUser) + ' ORDER BY DT.NAME, D.NAME';

 sqlPermDoc.Close;
 sqlPermDoc.SQL.Clear;
 sqlPermDoc.SQL.Add(sql);
 sqlPermDoc.Open;
 sqlPermDoc.FetchAll;

 //интерфейс
 btAdd.Enabled:= false;
 btDel.Enabled:= false;

 if sqlPermDoc.RecordCount = 0 then
  CurRule:= -1
 else
  btDel.Enabled:= true;

 if Length(SvDoc) <> 0 then
  btAdd.Enabled:= true;

 if not sqlPermDoc.Locate('id', CurRule, []) then CurRule:= sqlPermDocId.Value;
end;

//---------------------------------------------------------------------------

 //заполнить список пользователей
procedure TfmCategoryPerm.UserFill();
var pos: integer;
begin
 Finalize(SvUser);
 cbUser.Clear;

 sqlUser.Close;
 sqlUser.Open;
 sqlUser.FetchAll;

 SetLength(SvUser,sqlUser.RecordCount);
 sqlUser.First;
 pos:=0;
 while not sqlUser.Eof do
 begin
  SvUser[pos]:= sqlUserID.Value;
```

```
  cbUser.Items.Add(sqlUserFIO.Value);
  inc(pos);
  sqlUser.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать пользователя с индексом id. Если такого нет -1
function TfmCategoryPerm.UserStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvUser) do
  if SvUser[i] = id then
  begin
   Result:= true;
   cbUser.ItemIndex:= i;
   exit;
  end;

 cbUser.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

 //заполнить список документов
procedure TfmCategoryPerm.DocFill();
var pos: integer;
    sql: string;
begin
 Finalize(SvDoc);
 cbDoc.Clear;

 if CurUser = -1 then
 begin
   cbDoc.Enabled:= false;
   CurDoc:= -1;
   exit;
 end;

 cbDoc.Enabled:= true;

 //запрос
 sql:= 'SELECT D.ID, DT.NAME as "CatName", D.NAME as "DocName" FROM Document D INNER JOIN DocType DT ON
DT.ID=D.CATEGORY_ID LEFT JOIN DocPerm DP ON DP.DOCUMENT_ID=D.id AND DP.USER_ID=' + IntToStr(CurUser) + ' WHERE
DP.id IS NULL ORDER BY DT.NAME, D.Name';


 sqlDoc.Close;
 sqlDoc.SQL.Clear;
 sqlDoc.SQL.Add(sql);
 sqlDoc.Open;
 sqlDoc.FetchAll;

 SetLength(SvDoc,sqlDoc.RecordCount);
 sqlDoc.First;
 pos:=0;
 while not sqlDoc.Eof do
 begin
  SvDoc[pos]:= sqlDocID.Value;
  cbDoc.Items.Add(sqlDocCatName.Value + ' \ ' + sqlDocDocName.Value);
  inc(pos);
  sqlDoc.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать документ с индексом id. Если такого нет -1
function TfmCategoryPerm.DocStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvDoc) do
  if SvDoc[i] = id then
  begin
   Result:= true;
```

```
   cbDoc.ItemIndex:= i;
   exit;
  end;

 cbDoc.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.FormCreate(Sender: TObject);
begin
 CurRule:= -1;
 CurUser:= -1;
 CurDoc:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.FormActivate(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurRule:= -1;
 CurUser:= -1;
 CurDoc:= -1;

 sqlPermDoc.Close;
 sqlUser.Close;
 sqlDoc.Close;
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.btAddClick(Sender: TObject);
begin
 if CurUser = -1 then exit;
 if CurDoc = -1 then exit;

 fmS.AddDocumentPermition(CurDoc, CurUser);
 Fill();

 fmS.SaveLog('Предоставлен доступ пользователю №' + IntToStr(CurUser) + ' к документу №' + IntToStr(CurDoc)); //лог
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.btDelClick(Sender: TObject);
begin
 if CurRule = -1 then exit;

 if IDCANCEL = MessageBox(Handle,'Уверены что хотите удалить выбранное правило?','Внимание...',MB_OKCANCEL +
MB_ICONEXCLAMATION) then exit;

 fmS.ExecuteSQL('DELETE FROM DocPerm WHERE id=' + IntToStr(CurRule));

 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.dbgPermDocCellClick(Column: TColumn);
begin
 if sqlPermDoc.RecordCount = 0 then exit;
 CurRule:= sqlPermDocId.Value;
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.cbUserChange(Sender: TObject);
begin
 if cbUser.ItemIndex = -1  then exit;
 CurUser:= SvUser[cbUser.ItemIndex];
 sqlUser.Locate('id',CurUser,[]);
```

```
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmCategoryPerm.cbDocChange(Sender: TObject);
begin
 if cbDoc.ItemIndex = -1  then exit;
 CurDoc:= SvDoc[cbDoc.ItemIndex];
 sqlDoc.Locate('id',CurDoc,[]);
end;

//---------------------------------------------------------------------------

end.
```

Модуль DocType.pas

```
unit DocType;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, Grids, DBGrids, StdCtrls, ExtCtrls, DB, DBTables,
  IBCustomDataSet, IBQuery, IBDatabase;

type
  TfmDocType = class(TForm)
    Panel1: TPanel;
    btAdd: TButton;
    btChange: TButton;
    btDel: TButton;
    dbgCategory: TDBGrid;
    dsDocType: TDataSource;
    sqlList: TIBQuery;
    tList: TIBTransaction;
    sqlListID: TIntegerField;
    sqlListNAME: TIBStringField;

    procedure Fill();
    procedure btAddClick(Sender: TObject);
    procedure btChangeClick(Sender: TObject);
    procedure btDelClick(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure FormCreate(Sender: TObject);
    procedure dbgCategoryCellClick(Column: TColumn);
    procedure dbgCategoryDblClick(Sender: TObject);


  private
    { Private declarations }
  public
    CurDocType: integer;
  end;

var
  fmDocType: TfmDocType;

implementation

  uses SQLDM;

{$R *.dfm}


//---------------------------------------------------------------------------

  //Заполнение формы
procedure TfmDocType.Fill();
begin
 //Обновление данных запроса
 sqlList.Close;
 sqlList.Open;
 sqlList.FetchAll;
```

```
 if CurDocType <> -1 then
  if not sqlList.Locate('id',CurDocType,[]) then CurDocType:=-1;

 //кнопки
 if sqlList.RecordCount = 0 then
 begin
  btChange.Enabled:= false;
  btDel.Enabled:= false;
 end else
 begin
  btChange.Enabled:= true;
  btDel.Enabled:= true;
 end;
end;

//---------------------------------------------------------------------------

procedure TfmDocType.btAddClick(Sender: TObject);
var sName, sql: string;
begin
 if not InputQuery('Добавить категорию...','Введите название категории документации',sName) then exit;


 if sqlList.Locate('Name',sName,[loCaseInsensitive]) then
  if IDNO = MessageBox(Handle,'Такая категория уже числится в базе. Уверены что хотите внести её ещё раз?','Внимание...',MB_YESNO
+ MB_ICONQUESTION) then exit;

 //добавление записи
 sql:= 'INSERT INTO DocType (Name) VALUES ('#39 + sName  + #39')';
 fmS.ExecuteSQL(sql);
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmDocType.btChangeClick(Sender: TObject);
var sName, sql: string;
    Cur_id: integer;
begin
 if sqlList.RecordCount = 0 then exit; //блокировка

 //запоминаем текущие данные
 sName:=sqlListName.Value;
 Cur_id:= sqlListId.Value;
 if not InputQuery('Изменить название группы...','Введите новое название группы документации',sName) then exit;


 if sqlList.Locate('Name',sName,[loCaseInsensitive]) then
  if sqlListId.Value <> Cur_id then
   if IDNO = MessageBox(Handle,'Такая группа уже числится в списке. Уверены что хотите внести её ещё раз?','Внимание...',MB_YESNO
+ MB_ICONQUESTION) then exit;

 //изменение записи
 sql:= 'UPDATE DocType SET Name='#39 + sName  + #39' WHERE Id='+ IntToStr(Cur_id);
 fmS.ExecuteSQL(sql);
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmDocType.btDelClick(Sender: TObject);
var sql: string;
begin
 if IDCANCEL = MessageBox(Handle,'Уверены что хотите удалить выбранную группу?','Внимание...',MB_OKCANCEL +
MB_ICONEXCLAMATION) then exit;
 if sqlList.RecordCount = 0 then exit;  //блокировка

 sql:='DELETE FROM DocType WHERE id=' + sqlListId.AsString;
 fmS.ExecuteSQL(sql);
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmDocType.FormActivate(Sender: TObject);
begin
 Fill();
```

```
end;

//---------------------------------------------------------------------------

procedure TfmDocType.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
 sqlList.Close;
end;

//---------------------------------------------------------------------------

procedure TfmDocType.FormCreate(Sender: TObject);
begin
 CurDocType:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmDocType.dbgCategoryCellClick(Column: TColumn);
begin
 if sqlList.RecordCount = 0 then exit;
 CurDocType:= sqlListId.Value;
end;

//---------------------------------------------------------------------------

procedure TfmDocType.dbgCategoryDblClick(Sender: TObject);
begin
 btChangeClick(btChange);
end;

//---------------------------------------------------------------------------

end.
```

Модуль Log.pas

```
unit Log;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, IBDatabase, IBCustomDataSet, IBQuery, StdCtrls,
  Grids, DBGrids, ExtCtrls, SQLDM, Global;

type
  TfmLog = class(TForm)
    Panel1: TPanel;
    sqlLog: TIBQuery;
    tLog: TIBTransaction;
    dsLog: TDataSource;
    dbgLog: TDBGrid;
    lbGroups: TLabel;
    cbUser: TComboBox;
    sqlUser: TIBQuery;
    tUser: TIBTransaction;
    sqlUserID: TIntegerField;
    sqlUserFIO: TIBStringField;
    btClose: TButton;
    cbUserFiltr: TCheckBox;
    sqlLogFIO: TIBStringField;
    sqlLogU_LOGIN: TIBStringField;
    sqlLogID: TIntegerField;
    sqlLogACTION_DATE: TDateTimeField;
    sqlLogACTIONMES: TIBStringField;

    procedure FillLog();

    //пользователи
    procedure UserFill();
    function UserStandBy(id: integer): boolean;

    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
```

```
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btCloseClick(Sender: TObject);
    procedure cbUserChange(Sender: TObject);
    procedure cbUserFiltrClick(Sender: TObject);

  private
    SvUser: TDIntArr;
  public
    CurUser: integer;
  end;

var
  fmLog: TfmLog;

implementation

{$R *.dfm}

//---------------------------------------------------------------------------

//заполнение лога
procedure TfmLog.FillLog();
var sql, where_p: string;
begin
 //фильтр по пользователям
 where_p:= '';
 if cbUserFiltr.Checked then
  where_p:= ' WHERE A.USER_ID=' + IntToStr(CurUser);

  //запрос
 sql:= 'SELECT U.FIO, U.U_LOGIN, A.ID, A.ACTION_DATE, A.ACTIONMES FROM ACTIONLOG A INNER JOIN DBUSERS U ON
U.ID=A.USER_ID' + where_p + ' ORDER BY A.ACTION_DATE DESC';

 sqlLog.Close;
 sqlLog.SQL.Clear;
 sqlLog.SQL.Add(sql);
 sqlLog.Open;
 sqlLog.FetchAll;
end;

//---------------------------------------------------------------------------

 //заполнить список пользователей
procedure TfmLog.UserFill();
var pos: integer;
begin
 Finalize(SvUser);
 cbUser.Clear;

 sqlUser.Close;
 sqlUser.Open;
 sqlUser.FetchAll;

 SetLength(SvUser,sqlUser.RecordCount);
 sqlUser.First;
 pos:=0;
 while not sqlUser.Eof do
 begin
  SvUser[pos]:= sqlUserID.Value;
  cbUser.Items.Add(sqlUserFIO.Value);
  inc(pos);
  sqlUser.Next;
 end;
end;

//---------------------------------------------------------------------------

  //выбрать пользователя с индексом id. Если такого нет -1
function TfmLog.UserStandBy(id: integer): boolean;
var i: integer;
begin
 Result:= false;
 for i:=0 to High(SvUser) do
  if SvUser[i] = id then
  begin
   Result:= true;
   cbUser.ItemIndex:= i;
   exit;
```

```
  end;

 cbUser.ItemIndex:= -1; //запись не найдена
end;

//---------------------------------------------------------------------------

procedure TfmLog.FormCreate(Sender: TObject);
begin
 CurUser:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmLog.FormActivate(Sender: TObject);
begin
 //список пользователей
 UserFill();
 if Length(SvUser) <> 0 then
 begin
  if CurUser = -1 then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end
  else if not UserStandBy(CurUser) then
  begin
   cbUser.ItemIndex:= 0;
   CurUser:= SvUser[0];
  end;
 end;

 FillLog();
end;

//---------------------------------------------------------------------------

procedure TfmLog.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurUser:= -1;

 sqlLog.Close;
 sqlUser.Close;
end;

//---------------------------------------------------------------------------

procedure TfmLog.btCloseClick(Sender: TObject);
begin
 Close();
end;

//---------------------------------------------------------------------------


procedure TfmLog.cbUserChange(Sender: TObject);
begin
 if cbUser.ItemIndex = -1  then exit;
 CurUser:= SvUser[cbUser.ItemIndex];
 sqlUser.Locate('id',CurUser,[]);

 FillLog();
end;

//---------------------------------------------------------------------------

procedure TfmLog.cbUserFiltrClick(Sender: TObject);
begin
 FillLog();
end;

//---------------------------------------------------------------------------

end.
```

Модуль UserList.pas

```
unit UserList;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, DB, IBDatabase, IBCustomDataSet, IBQuery, StdCtrls,
  Grids, DBGrids, ExtCtrls, SQLDM, Global;

type
  TfmUserList = class(TForm)
    Panel1: TPanel;
    btAdd: TButton;
    btEdit: TButton;
    btDel: TButton;
    sqlUser: TIBQuery;
    tUser: TIBTransaction;
    dsUser: TDataSource;
    dbgUser: TDBGrid;
    sqlUserkalk_state: TStringField;
    sqlUserID: TIntegerField;
    sqlUserFIO: TIBStringField;
    sqlUserSTATE: TIntegerField;
    sqlUserGroupName: TIBStringField;
    sqlUserU_LOGIN: TIBStringField;

    procedure Fill();

    procedure sqlUserCalcFields(DataSet: TDataSet);
    procedure FormCreate(Sender: TObject);
    procedure FormActivate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure btAddClick(Sender: TObject);
    procedure btEditClick(Sender: TObject);
    procedure btDelClick(Sender: TObject);
    procedure dbgUserCellClick(Column: TColumn);

  private
    { Private declarations }
  public
    CurUser: integer;
  end;

var
  fmUserList: TfmUserList;

implementation

{$R *.dfm}

 uses AddUser;

//---------------------------------------------------------------------------

procedure TfmUserList.Fill();
begin
 sqlUser.Close;
 sqlUser.Open;
 sqlUser.FetchAll;

 if sqlUser.RecordCount = 0 then
 begin
  btEdit.Enabled:= false;
  btDel.Enabled:= false;
  CurUser:= -1;
  exit;
 end;

 btEdit.Enabled:= true;
 btDel.Enabled:= true;

 if not sqlUser.Locate('id', CurUser, []) then CurUser:= sqlUserId.Value;
end;

//---------------------------------------------------------------------------

procedure TfmUserList.sqlUserCalcFields(DataSet: TDataSet);
```

```
begin
 if sqlUserSTATE.Value = 1 then sqlUserkalk_state.Value:= 'активен'
 else sqlUserkalk_state.Value:= 'заблокирован';
end;

//---------------------------------------------------------------------------

procedure TfmUserList.FormCreate(Sender: TObject);
begin
 CurUser:= -1;
end;

//---------------------------------------------------------------------------

procedure TfmUserList.FormActivate(Sender: TObject);
begin
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmUserList.FormClose(Sender: TObject; var Action: TCloseAction);
begin
 CurUser:= -1;
 sqlUser.Close;
end;

//---------------------------------------------------------------------------

procedure TfmUserList.btAddClick(Sender: TObject);
begin
 fmAddUser.RegSt:= rsNew;
 fmAddUser.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmUserList.btEditClick(Sender: TObject);
begin
 if CurUser = -1 then exit;

 fmAddUser.UserID:= CurUser;
 fmAddUser.RegSt:= rsEdit;

 fmAddUser.ShowModal;
 Fill();
end;

//---------------------------------------------------------------------------

procedure TfmUserList.btDelClick(Sender: TObject);
begin
 if CurUser = -1 then exit;

 if CurUser = CurrentUserID then
 begin
  MessageBox(handle, 'Вы не можете удалить свою собственную учётную запись','Ошибка!', MB_OK + MB_ICONEXCLAMATION);
  exit;
 end;

 if fmS.IsActivUser(CurUser) then
 begin
  MessageBox(handle, 'Данный пользователь сейчас работает с базой. Нельзя удалять активного пользователя','Ошибка!', MB_OK +
MB_ICONEXCLAMATION);
  exit;
 end;

 if IDCANCEL = MessageBox(Handle,'Уверены что хотите удалить выбранного пользователя?','Внимание...',MB_OKCANCEL +
MB_ICONEXCLAMATION) then exit;

 fmS.ExecuteSQL('UPDATE DBUSERS SET State=0 WHERE id=' + IntToStr(CurUser));

 Fill();
end;

//---------------------------------------------------------------------------
```

```
procedure TfmUserList.dbgUserCellClick(Column: TColumn);
begin
 if sqlUser.RecordCount = 0 then exit;
 CurUser:= sqlUserId.Value;
end;

//---------------------------------------------------------------------------

end.
```

Модуль About.pas

```
unit About;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls;

type
  TfmAbout = class(TForm)
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  fmAbout: TfmAbout;

implementation

{$R *.dfm}

end.
```

Модуль SQLDM.pas

```
unit SQLDM;

interface

uses
  SysUtils, Classes, DB, IBCustomDataSet, IBQuery, IBDatabase, Global;

type
  TfmS = class(TDataModule)
    DB: TIBDatabase;
    tData: TIBTransaction;
    sqlData: TIBQuery;

    procedure OpenSQL(sql: string);
    procedure ExecuteSQL(sql: string);

    {вспомогательные функции}
      //добавляет в строку where или значение операнда (and, or) в зависимости от её состояния
    procedure CorrectWhere(var Where_str: string; Operand: string);

    //==============

    //работа с пользователями
    function ActivateUser(UserID: integer): boolean;
    procedure DeactivateUser(UserID: integer);
    procedure ClearActivUsers(UserID: integer);
    function IsActivUser(UserID: integer): boolean;
```

```
    //работа с документами
    function DocStatus(Status: integer): integer;
    procedure AddUserInStatus(DocId: integer; UserID: integer);
    procedure DelUserFromStatus(DocId: integer);
    function GetLastDocumentByName(Name: string): integer;

    //работа с правами пользователей
    function IsActionPermition(Action: string): boolean;          //проверка прав на действие
    function IsDocumentPermition(DocID: integer): boolean;    //проверка прав на работу с документом
    procedure AddDocumentPermition(DocID: integer; UserID: integer);    //добавление прав на работу с документом


    //архив
    procedure SaveToArhiv(DocID: integer);
    procedure LoadFromArhiv(ArhivID, DocID: integer);
    function IsArhiv(DocID: integer): boolean;


    //лог
    procedure SaveLog(Mess: string);


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


//Правило:
//  Статус пользователя: 0 - user, 1 - admin
//  Статус документа: 0 - закрыт, 1 - используется


var
  fmS: TfmS;

implementation

{$R *.dfm}

//---------------------------------------------------------------------------

procedure TfmS.OpenSQL(sql: string);
begin
 sqlData.Close;
 sqlData.SQL.Clear;
 sqlData.SQL.Add(sql);
 sqlData.Open;
 sqlData.FetchAll;
end;


//---------------------------------------------------------------------------

procedure TfmS.ExecuteSQL(sql: string);
begin
 if tData.Active then tData.Rollback;
 tData.StartTransaction;
 try
  sqlData.Close;
  sqlData.SQL.Clear;
  sqlData.SQL.Add(sql);
  sqlData.ExecSQL;
  tData.Commit;
 except
  tData.Rollback;
 raise;
 end;
end;

//---------------------------------------------------------------------------

    //добавляет в строку where или значение операнда (and, or) в зависимости от её состояния
procedure TfmS.CorrectWhere(var Where_str: string; Operand: string);
begin
 if Where_str = '' then Where_str:= Where_str + ' WHERE'
 else Where_str:= Where_str + ' ' + Operand;
```

```
end;

//=============================================================================

  //пытается добавить пользователя в список активных. Если такой уже есь вернёт false
function TfmS.ActivateUser(UserID: integer): boolean;
begin
 Result:= false;
 if IsActivUser(UserID) then exit;

 ExecuteSQL('INSERT INTO ActivUser (user_id, document_id, state) VALUES (' + IntToStr(UserID) + ', 0, 0)');
 Result:= true;
end;

//-----------------------------------------------------------------------------

  //удаляет пользователя из списка активных
procedure TfmS.DeactivateUser(UserID: integer);
begin
 ExecuteSQL('DELETE FROM ActivUser WHERE user_id=' + IntToStr(UserID) + ' AND document_id=0 AND state=0');
end;

//-----------------------------------------------------------------------------

  //очищает список активных пользователей
  //в параметре передан пользователь которого нужно оставить
procedure TfmS.ClearActivUsers(UserID: integer);
begin
 ExecuteSQL('DELETE FROM ActivUser WHERE id_user<>' + IntToStr(UserID) + ' AND document_id=0 AND state=0');
end;

//-----------------------------------------------------------------------------

  //проверяет активен ли пользователь
function TfmS.IsActivUser(UserID: integer): boolean;
begin
 Result:= false;

 OpenSQL('SELECT user_id FROM ActivUser WHERE user_id=' + IntToStr(UserID) + ' AND document_id=0 AND state=0');
 if sqlData.RecordCount = 0 then exit;

 Result:= true;
end;


//-----------------------------------------------------------------------------

  //статус документа DocID
  //0 - закрыт  1 - открыт  2 - используется (т.е. редакт др пользователем)
function TfmS.DocStatus(Status: integer): integer;
begin
 Result:= 0;
 if Status = 0 then exit;

 if Status = CurrentUserID then Result:= 1
 else Result:= 2;
end;

//-----------------------------------------------------------------------------

  //установка в поле статуса идентификатора пользователя
procedure TfmS.AddUserInStatus(DocId: integer; UserID: integer);
begin
 ExecuteSQL('UPDATE Document SET State=' + IntToStr(UserID) + ' WHERE id=' + IntToStr(DocId));
end;

//-----------------------------------------------------------------------------

  //очистка поля статуса при закрытии документа
procedure TfmS.DelUserFromStatus(DocId: integer);
begin
  ExecuteSQL('UPDATE Document SET State=0 WHERE id=' + IntToStr(DocId));
end;

//-----------------------------------------------------------------------------

function TfmS.GetLastDocumentByName(Name: string): integer;
begin
```

```
 Result:= -1;

 OpenSQL('SELECT id FROM DOCUMENT WHERE NAME='#39 + Name + #39' ORDER BY id DESC' );
 if sqlData.RecordCount = 0 then exit;

 sqlData.First;
 Result:= sqlData.FieldByName('id').AsInteger;
end;

//---------------------------------------------------------------------------

//проверка прав на действие
function TfmS.IsActionPermition(Action: string): boolean;
begin
 Result:= false;

 OpenSQL('SELECT P.id FROM ActionPerm P INNER JOIN ActionType A ON A.ID=P.ACTION_ID WHERE P.GROUP_ID=' +
IntToStr(CurrentUserType) + ' AND A.ACTNAME='#39 + Action + #39);
 if sqlData.RecordCount = 0 then exit;

 Result:= true;
end;

//---------------------------------------------------------------------------

//проверка прав на работу с категорией
function TfmS.IsDocumentPermition(DocID: integer): boolean;
begin
 Result:= false;

 //админу можно всё
 if CurrentUserType = 1 then
 begin
  Result:= true;
  exit;
 end;

 OpenSQL('SELECT id FROM DocPerm WHERE DOCUMENT_ID=' + IntToStr(DocID) + ' AND USER_ID=' + IntToStr(CurrentUserID));
 if sqlData.RecordCount = 0 then exit;

 Result:= true;
end;

//---------------------------------------------------------------------------

procedure TfmS.AddDocumentPermition(DocID: integer; UserID: integer);    //добавление прав на работу с документом
begin
 fmS.ExecuteSQL('INSERT INTO DocPerm (USER_ID, DOCUMENT_ID) VALUES (' + IntToStr(UserID) + ',' + IntToStr(DocID) + ')');
end;

//---------------------------------------------------------------------------

//сохранить архив
procedure TfmS.SaveToArhiv(DocID: integer);
var sql: string;
begin
 sql:= 'INSERT INTO DOCUMENTARHIV (SAVE_DATE, DOC, DOCUMENT_ID) SELECT cast('#39'now'#39' as timestamp), DOC, ID
FROM DOCUMENT WHERE ID = ' + IntToStr(DocID);
 ExecuteSQL(sql);

 //лог
 SaveLog('Создана архивная копия документа №' + IntToStr(DocID));
end;

//---------------------------------------------------------------------------

//загрузить из архива
procedure TfmS.LoadFromArhiv(ArhivID, DocID: integer);
var sql: string;
begin
 //первым шагом идёт сохранение текущей версии документа в архив
 SaveToArhiv(DocID);

 //затем восстанавливаем запрошенную версию из архива
 sql:= 'UPDATE DOCUMENT SET DOC=(SELECT DOC FROM DOCUMENTARHIV WHERE ID=' + IntToStr(ArhivID) + ') WHERE ID='
+ IntToStr(DocID);
 ExecuteSQL(sql);
```

```
 SaveLog('Восстановлена прежняя версия №' + IntToStr(ArhivID) + ' документа №' + IntToStr(DocID));
end;

//---------------------------------------------------------------------------

//есть ли у документа архивные копии
function TfmS.IsArhiv(DocID: integer): boolean;
begin
 Result:= false;

 OpenSQL('SELECT id FROM DOCUMENTARHIV WHERE DOCUMENT_ID=' + IntToStr(DocID));
 if sqlData.RecordCount = 0 then exit;

 Result:= true;
end;

//---------------------------------------------------------------------------

//запись в лог
procedure TfmS.SaveLog(Mess: string);
var sql: string;
begin
 sql:= 'INSERT INTO ACTIONLOG (ACTION_DATE, USER_ID, ACTIONMES) VALUES (cast('#39'now'#39' as timestamp), ' +
IntToStr(CurrentUserID) + ', '#39 + Mess + #39')';
 ExecuteSQL(sql);
end;

//---------------------------------------------------------------------------

end.
```

Модуль Global.pas

```
unit Global;

interface

  uses Windows, SysUtils, Graphics, Classes;

  type

    TDStrArr = array of string;
    TDIntArr = array of integer;  //индексный массив

    TRegStatus = (rsNone, rsNew, rsEdit);
    TStudyTip = (stKyrs,stSeminar);

    //------- CFG типы

    //настройки подключения к базе данных
    TRDatabase = record
     ServerName: string;
     Username: string;
     Password: string;
    end;


    TRCFG = record    //общая запись
     DBConfig: TRDatabase;
    end;

    //----------

    TVal = class   //класс содержащий число. Требуется для связи TTreeViev с базой данных
     Key: integer;
    end;

    TNodeRec = record //id и текстов. строка
     id: integer;
     Descr: string;
    end;

    TDNodeRec = array of TNodeRec;  //список узлов

    //-----------
```

```
 //возвращает фамилию с инициалами
 function MakeShortName(aFam,aName,aOtch: string): string;

 //возвращает в виде строки разницу во времени
 function TimeDiferent(TimeStart, TimeFinish: TDateTime): string;

 //замена подстрок
 function str_replace(str, sub1, sub2: string): string;

 //поиск слов в строке (разделитель пробел) Вернёт -1 если не нащёл
 function str_findU(str: string; sub: string; ind: integer): boolean;

 //делим строку на слова
 function ExplodeWodr(Str: string): TDStrArr;

 //делим строку на массив подстрок по символу sim
 function Explode(sim: char; Str: string): TDStrArr;

 //сливаем строку с разделителем в виде символа sim
 function Implode(sim: char; Arr: TDStrArr): string;

 //удаление из строки всех ненужных символов
 procedure PrepareStr(Simb: string; var Str: string);

 //Очистка строк от лишних пробелов
 function ProbelNorm(Str: string): string;


///методы обработки строковых массивов TDStrArr

    //добавляет Str в конец массива Arr
 procedure AddValSA(var Arr: TDStrArr; Str: string);

    //ищет Str в массиве Arr  (-1 если строка не найдено)
 function FindInSA(var Arr: TDStrArr; Str: string): integer;

    //удаляет элемент в позиции Pos
 procedure DelPosSA(var Arr: TDStrArr; Pos: integer);



///----методы обраб индексных массивов TDIntArr

 //добавляет Value в конец массива Arr
 procedure AddValIA(var Arr: TDIntArr; Value: integer);

 //удаляет из массива Arr число по его позиции Pos
 procedure DelPosIA(var Arr: TDIntArr; Pos: integer);

 //позвращает позицию числа Value в массиве Arr (-1 если число не найдено)
 function InArr(var Arr: TDIntArr; Value: integer):integer;

   //полное копирование индексного массива из aSourse в aDest
 procedure AssignIntArr(var aDest: TDIntArr; aSourse: TDIntArr);

   //проверяет массивы не эквивалентность
 function IsEcvIA(var Arr1: TDIntArr; var Arr2: TDIntArr): boolean;


 {Методы для обработки CFG файла}
  //Сохранения
 procedure SaveDBConfig(); //сохраняем настройки БД
 procedure SaveCFG(); //сохраняем параметры

  //Загрузки
 procedure LoadDBConfig(); //загружаем настройки БД
 procedure LoadCFG(); //загружаем параметры

  //очистка cfg файла
 procedure ClrCFG();
```

```
  var Dir: string;  //директория программы
      CurrentUserID : integer;   //текущий пользователь
      CurrentUserType: integer; //статус пользователя
      StartT, EndT: TDateTime;
      CFG_File: textfile;
      CFG: TRCFG;

implementation

//---------------------------------------------------------------------------

  {возвращает фамилию с инициалами}
function MakeShortName(aFam,aName,aOtch: string): string;
begin
 Result:= aFam;
 if aName <> '' then Result:= Result + ' ' + aName[1] + '.';
 if aOtch <> '' then Result:= Result + ' ' + aOtch[1] + '.';
end;

//---------------------------------------------------------------------------

   //возвращает в виде строки разницу во времени
function TimeDiferent(TimeStart, TimeFinish: TDateTime): string;
var H,M,S,ms: word;
    DifTime: TDateTime;
begin
 DifTime:= TimeFinish - TimeStart;
 DecodeTime(DifTime,H,M,S,ms);
 Result:= 'Ч: ' + IntToStr(H) + '  M: ' + IntToStr(M) + ' сек: ' + IntToStr(S) + ' мс: ' + IntToStr(ms);
end;

//---------------------------------------------------------------------------

function str_replace(str, sub1, sub2: string): string;
var
  aPos: Integer;
  rslt: string;
begin
  aPos := Pos(sub1, str);
  rslt := '';
  while (aPos <> 0) do
  begin
    rslt := rslt + Copy(str, 1, aPos - 1) + sub2;
    Delete(str, 1, aPos + Length(sub1) - 1);
    aPos := Pos(sub1, str);
  end;
  Result := rslt + str;
end;

//---------------------------------------------------------------------------

     //поиск слов в строке (разделитель пробел) Вернёт -1 если не нащёл
function str_findU(str: string; sub: string; ind: integer): boolean;
var Arr: TDStrArr;
    i: integer;
begin
 str:= Copy(str,ind,Length(str)-ind);
 str:= AnsiUpperCase(str);
 sub:= AnsiUpperCase(sub);

 Arr:= ExplodeWodr(sub);
 for i:=0 to High(Arr) do
  if Pos(Arr[i],str) > 0 then
  begin
   Result:= true;
   exit;
  end;

 Result:= false;
end;

//---------------------------------------------------------------------------

     //делим строку на слова
function ExplodeWodr(Str: string): TDStrArr;
var Count,i,ArrPos: integer;
    NewWord: boolean;
    Stroka: string;
```

```
begin
 Finalize(Result);
 Count:=0;
 for i:=1 to Length(Str) do
 begin
  if Str[i] = ' ' then NewWord:= false
  else NewWord:= true;

  if NewWord then Stroka:= Stroka + Str[i]
  else
   if Stroka <> '' then
   begin
    ArrPos:= Count;
    inc(Count);
    SetLength(Result, Count);
    Result[ArrPos]:= Stroka;
    Stroka:= '';
   end;
 end;

 if Stroka <> '' then
 begin
  ArrPos:= Count;
  inc(Count);
  SetLength(Result, Count);
  Result[ArrPos]:= Stroka;
  Stroka:= '';
 end;
end;

//---------------------------------------------------------------------------

    //делим строку на массив подстрок по символу sim
function Explode(sim: char; Str: string): TDStrArr;
var i: integer;
   CurS: string;
begin
 CurS:='';
 Finalize(Result);
 for i:=1 to Length(Str) do
  if Str[i] = sim then
  begin
   AddValSA(Result,CurS);
   CurS:='';
  end else CurS:= CurS + Str[i];
 AddValSA(Result,CurS);
end;

//---------------------------------------------------------------------------

    //сливаем строку с разделителем в виде символа sim
function Implode(sim: char; Arr: TDStrArr): string;
var i: integer;
begin
 Result:= '';
 if Length(Arr) = 0 then exit;

 Result:= Arr[0];

 for i:=1 to High(Arr) do Result:= Result + sim + Arr[i];
end;

//---------------------------------------------------------------------------

    //удаление из строки всех ненужных символов
procedure PrepareStr(Simb: string; var Str: string);
begin
 //
end;

//---------------------------------------------------------------------------

    //Очистка строк от лишних пробелов
function ProbelNorm(Str: string): string;
var i,p,L: integer;
begin
 Result:='';
 L:= Length(Str);
```

```
 p:=1;
 for i:=1 to L do
  if Str[i] <> ' ' then
  begin
   p:=i;
   break;
  end;

 for i:=p to L-1 do
  if (Str[i] <> ' ') or (Str[i+1] <> ' ') then Result:= Result + Str[i];
 if (L<>0) and (Str[L]<>' ') then Result:= Result + Str[L];
end;

//========== {методы обработки строковых массивов TDStrArr} ==================

        //добавляет Value в конец массива Arr
procedure AddValSA(var Arr: TDStrArr; Str: string);
var l: integer;
begin
 l:= Length(Arr);
 SetLength(Arr, l+1);
 Arr[l]:= Str;
end;

//---------------------------------------------------------------------------

        //ищет Str в массиве Arr  (-1 если строка не найдено)
function FindInSA(var Arr: TDStrArr; Str: string): integer;
var i: integer;
begin
 Result:= -1;
 for i:=0 to High(Arr) do
  if Arr[i] = Str then
  begin
   Result:= i;
   exit;
  end;
end;

//---------------------------------------------------------------------------

        //удаляет элемент в позиции Pos
procedure DelPosSA(var Arr: TDStrArr; Pos: integer);
var i: integer;
begin
 if (Pos < 0) or (Pos > Length(Arr)-1) then exit;

 for i:=Pos to High(Arr)-1 do Arr[i]:= Arr[i+1];
 SetLength(Arr, Length(Arr)-1);
end;

//========== {методы обраб индексных массивов TDIntArr} ======================

    //добавляет Value в конец массива Arr
procedure AddValIA(var Arr: TDIntArr; Value: integer);
var l: integer;
begin
 l:= Length(Arr);
 SetLength(Arr, l+1);
 Arr[l]:= Value;
end;

//---------------------------------------------------------------------------

    //удаляет из массива Arr число по его позиции Pos
procedure DelPosIA(var Arr: TDIntArr; Pos: integer);
var i: integer;
begin
 if Pos > Length(Arr)-1 then exit;
 for i:=Pos to High(Arr)-1 do Arr[i]:= Arr[i+1];
 SetLength(Arr, Length(Arr)-1);
end;

//---------------------------------------------------------------------------

    //позвращает позицию числа Value в массиве Arr (-1 если число не найдено)
function InArr(var Arr: TDIntArr; Value: integer):integer;
var i: integer;
```

```
begin
 for i:=0 to High(Arr) do
  if Arr[i] = Value then
  begin
   Result:= i;
   exit;
  end;
 Result:= -1;
end;

//---------------------------------------------------------------------------

       //полное копирование индексного массива из aSourse в aDest
procedure AssignIntArr(var aDest: TDIntArr; aSourse: TDIntArr);
var i: integer;
begin
 SetLength(aDest,Length(aSourse));
 for i:=0 to High(aSourse) do aDest[i]:= aSourse[i];
end;

//---------------------------------------------------------------------------

       //проверяет массивы не эквивалентность
function IsEcvIA(var Arr1: TDIntArr; var Arr2: TDIntArr): boolean;
var i,j: integer;
    find: boolean;
begin
 Result:= false;
 if Length(Arr1) <> Length(Arr2) then exit;
 for i:=0 to High(Arr1) do
 begin
  find:= false;
  for j:=0 to High(Arr2) do
   if Arr1[i] = Arr2[j] then
   begin
    find:= true;
    break;
   end;
  if not find then exit;
 end;
 Result:= true;
end;


//================== {Методы для обработки CFG файла} ==========================

procedure SaveDBConfig(); //сохраняем настройки БД
begin
 writeln(CFG_File,'');
 writeln(CFG_File,'[db]');
 with CFG.DBConfig do
 begin
  writeln(CFG_File,ServerName);
  writeln(CFG_File,UserName);
  writeln(CFG_File,Password);
 end;
end;

//---------------------------------------------------------------------------

procedure SaveCFG(); //сохраняем параметры
begin
 AssignFile(CFG_File, Dir + 'config.cfg');
 Rewrite(CFG_File);

 SaveDBConfig(); //настройки БД

 CloseFile(CFG_File);
end;

//---------------------------------------------------------------------------

procedure LoadDBConfig(); //загружаем настройки БД
var s: string;
begin
 readln(CFG_File,s);
 while (s <> '[db]') and not Eof(CFG_File) do readln(CFG_File,s);
 with CFG.DBConfig do
```

```
 begin
  readln(CFG_File,ServerName);
  readln(CFG_File,UserName);
  readln(CFG_File,Password);
 end;
end;

//---------------------------------------------------------------------------

procedure LoadCFG(); //загружаем параметры
begin
 if not FileExists(Dir + 'config.cfg') then
 begin
  ClrCFG();
  exit;
 end;

 AssignFile(CFG_File, Dir + 'config.cfg');
 Reset(CFG_File);

 LoadDBConfig(); //настройки БД

 CloseFile(CFG_File);
end;

//---------------------------------------------------------------------------

      //очистка cfg файла
procedure ClrCFG();
begin
 //настройки БД
 with CFG.DBConfig do
 begin
  ServerName:='';
  Username:='';
  Password:='';
 end;
end;

//---------------------------------------------------------------------------

initialization
 CurrentUserID:= 0;
 CurrentUserType:= 0;

end.
```