ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«БЕЛГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ»
(Н И У « Б е л Г У »)

ИНСТИТУТ ИНЖЕНЕРНЫХ ТЕХНОЛОГИЙ И ЕСТЕСТВЕННЫХ НАУК

Кафедра прикладной информатики и информационных технологий

РАЗРАБОТКА АВТОМАТИЗИРОВАННОЙ ПОДСИСТЕМЫ УЧЕТА И ПРОДАЖИ МЕДИЦИНСКИХ ТОВАРОВ

Выпускная квалификационная работа бакалавра

очной формы обучения
направления подготовки 09.03.03 Прикладная информатика

4 курса группы 07001204
Полякова Алексея Олеговича

Научный руководитель
к. ф.-м. н., доцент Кузьмичева Т.Г.

БЕЛГОРОД 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На сегодняшний день сложно обойтись без специальных программных средств, которые позволяют не только сэкономить рабочее время сотрудников при выполнении рутинных операций, а также минимизировать количество ошибок, связанных с человеческим фактором. В настоящее время большинство аптек и лечебно-профилактических учреждений ведут контроль над движением аптечных товаров, не используя никаких специализированных программных средств.

Целью данной работы является разработка и реализация программного продукта, связанного с ведением учета лекарственных средств и изделий медицинского назначения в аптеках и лечебно-профилактических учреждений в городе Белгороде.

Для решения поставленной цели необходимо решить следующие задачи:

– изучить организационную структуру предприятия;

– выявить недостатки существующей организации обработки информации, которые определяют необходимость разработки данной работы;

– обосновать принятие проектных решений;

– спроектировать и разработать базу данных;

– создать программный продукт, отвечающий всем поставленным перед ним требованиям. А именно: централизованное хранение данных об аптечных товарах, предоставление информации о лекарственных средствах и изделиях медицинского назначения, имеющихся в учреждении на любой момент времени в денежном и количественном выражении, формирование необходимые в процессе работы аптеки отчетов, и запросов. Подсистема должна иметь простой и понятный интерфейс;

– протестировать созданный продукт;

– оценить эффективность работы предприятия внедряемым программным продуктом.

Объектом исследования является ООО «АРТФАРМ».

Предметом исследования являются программно-технические средства проектирования и разработки автоматизированной информационной подсистемы.

Данная работа состоит из трех разделов, в которых рассматриваются все этапы проектирования и разработки подсистемы автоматизации учета и продажи медицинских товаров.

Первый раздел посвящен технико-экономической характеристике предприятия. В нем будет рассмотрена и проанализирована организационная структура предприятия. На основе анализа разработаны модели работы подразделений организации и их взаимодействие. Также будут проанализированы уже существующие специализированные программные продукты учета и продажи медицинских товаров. Будут выявлены основные характеристики, которыми должен обладать разрабатываемый продукт.

Второй раздел работы посвящен обоснованию проектных решений. Будут рассмотрены и обоснованы решения по техническому, программному и информационному обеспечению.

Третий раздел - проектная часть, будет посвящена проектированию и разработке подсистемы учета и продажи медицинских товаров. Будут построены функциональные модели подсистемы, выделены соответствующие модули. Разработка будет выполняться в инструментальном программном средстве «C++ Builder 6». В конце проектной части рассмотрим целесообразность разработки, с экономической точки зрения.

Выпускная квалификационная работа содержит 66 листов, в которые входят 3 таблицы и 53 рисунка.

1 ТЕХНИКО-ЭКОНОМИЧЕСКАЯ ХАРАКТЕРИСТИКА ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Общая характеристика структуры и деятельности организации

В качестве рассматриваемой организации будет выступать ООО «АРТФАРМ». Данная организация является одной из крупнейших фармацевтических сетей в городе Белгороде.

Во-первых, форма хозяйственной деятельности - общество с ограниченной ответственностью (общепринятое сокращение - ООО). Это значит, что это учрежденное одним или несколькими юридическими и/или физическими лицами хозяйственное общество, уставный капитал которого разделён на доли. Участники общества не отвечают по его обязательствам и несут риск убытков, связанных с деятельностью общества, в пределах стоимости принадлежащих им долей или акций в уставном капитале общества[3].

Во-вторых, так как организация является сетью, то вся деятельность рассредоточена в нескольких магазинах. Каждый аптечный магазин имеет широкий ассортимент и большую площадь. Здесь существует торговля через прилавок, и в торговом зале находится продавец-консультант. Особенность аптечного магазина - пропаганда здорового образа жизни и продажа соответствующих товаров (витамины, БАДы, лечебные напитки и минеральные воды и т.д.).

В-третьих, функции данной организации и персонала:

– логистическая (прием, хранение, управление товарными запасами);

– производственная (прием рецептов, контроль и отпуск лекарственных средств по рецептам);

– информационная (обеспечение населения и врачей информацией о лекарственных средствах);

– маркетинговая (формирование и осуществление ассортиментной и ценовой политики);

– медицинская (оказание первой доврачебной помощи при необходимости).

Данная организация, как и любая другая имеет свою организационную структуру. Организационная структура - это совокупность подразделений организации и их взаимосвязей, в рамках которой между подразделениями распределяются управленческие задачи, определяются полномочия и ответственность руководителей и должностных лиц. Организационная структура устанавливается исходя из целей деятельности и необходимых для достижения этих целей подразделений, выполняющих функции, составляющие бизнес-процессы организации. Как правило, она отображается в виде графической схемы, элементами которой являются иерархически упорядоченные организационные единицы (подразделения, должностные позиции)[11]. На рисунке 1.1 изображена организационная структура ООО «АРТФАРМ».

Рисунок 1.1 - Организационная структура предприятия

Далее рассмотрели распределение управленческих функций.

Руководство предприятия: во главе каждой организации стоит директор. Директор коммерческого предприятия является доверенным лицом учредителей, единоначальником и отвечает за результаты всей деятельности предприятия.

Директор определяет пути и методы выполнения задач, поставленных перед предприятием, обеспечивает получение необходимых материально-технических ресурсов, организует труд на производстве, руководит работой по внедрению новой техники и технологии. Директор несет ответственность за выполнение предприятием обязательств перед клиентами. На предприятии постоянно проводится работа по улучшению условий и организации труда.

Отдел маркетинга: важным органом управленческого аппарата предприятия является отдел маркетинга. На основании указаний директора он разрабатывает перспективные и текущие планы деятельности предприятия, осуществляет руководство составлением планов, координирует работу других отделов, контролирует выполнение планов, организует и осуществляет анализ их выполнения. Маркетинговый отдел помогает коллективу в мобилизации сил и средств, для решения основных задач, стоящих перед предприятием. С этой целью маркетинговый отдел под руководством директора конкретизирует положения плана и доводит их до каждого отдела. Кроме того, занимаясь анализом хозяйственной деятельности предприятия, он указывает коллективу работников на отстающие участки и нерешенные задачи. Для контроля над выполнением плана обычно организуются статистический учет и отчетность на всех участках предприятия.

Отдел кадров: работой по комплектованию кадров предприятия руководит отдел кадров, а также смежный с ним отдел по подготовке кадров. Кадровый отдел организует работу на предприятии и проводит систему мероприятий, направленных на повышение квалификации работников. Штат аптеки делится на три группы: руководящие работники, специалисты,

производственный персонал. Численность персонала определяется аптекой и зависит от типа аптеки и объема ее работы.

Бухгалтерия: осуществляет учет средств предприятия, контролирует сохранность имущества, определяет себестоимость продукции, контролирует целесообразность и законность финансовых операций предприятия, и соблюдение финансовой дисциплины. Финансовым хозяйством на предприятиях небольшого размера ведает бухгалтерия, а на крупных предприятиях - специальный финансовый отдел, подчиненный директору.

Отдел снабжения: контролирует все необходимые материалы, и препараты.

Сбыт готовой продукции на крупных предприятиях с большим количеством потребителей осуществляется отделом сбыта, который ведает реализацией готовой продукции. На предприятиях с ограниченным кругом потребителей создается единый отдел снабжения и сбыта. В ряде случаев отдел сбыта объединяется с финансовым отделом.

1.2 Анализ существующих подсистем учета и продаж медицинских товаров внутри организаций

При организации деятельности аптеки необходимо принимать во внимание такие сложные задачи, как работа с поставщиками, заказ товара, определение товаров, которые могут вскоре закончиться, продажа лекарств покупателям и т.д. Эти отдельные задачи в аптеке выполняются примерно в одно время, поэтому следить за правильной работой системы становится сложнее. Анализ программного обеспечения свидетельствует о том, что предпочтения отдается стандартным приложениям Microsoft Office, а именно Microsoft Word - где все документы и отчеты создаются вручную или с помощью конструктора, Microsoft Excel - с его помощью производятся расчеты, а также он может выступать в роли небольшой базы данных, и Microsoft Access - база данных препаратов и их стоимости.

Необходимо отметить, что есть и специализированное программное обеспечение:

1) Программа «экми-мастер (аптека)» предназначена для аптек и аптечных объединений и включает в себя полный комплекс учета – движение товара, бухгалтерский учет, налоговый учет, расчет зарплаты, учет кадров. Учет розничных продаж в аптеке реализован с использованием фискального регистратора или принтера чеков[5].

Основные функции, реализуемые программой «ЭКМИ-МАСТЕР (аптека)»:

– движение товара;

– формирование справочника накладных по приходу товара от поставщика;

– формирование счетов;

– переоценка товара;

– розничная реализация товара;

– списание товара;

– количественный учет;

– журнал сроков годности;

Преимуществами программы «экми-мастер (аптека)» являются простота и логичность интерфейса, полное соответствие действующему законодательству, минимальные требования к вычислительной технике.

Недостатками данной программы является ее высокая стоимость, и сложность в первоначальной настройке.

2) 1С: Розница 8. Аптека - предназначена для работы со специфическим ассортиментом (лекарственные средства, товары медицинского назначения) и бизнес-процессами (серийный учет лекарственных средств, контроль фальсификатов и сроков годности, контроль правил ценообразования и т.д.) аптек, как в варианте одиночной аптеки, так и сети аптек[6].

Основные функциональные возможности отраслевого решения:

– оформление прихода лекарственных средств от контрагента на склады аптек;

– оформление перемещения лекарственных средств между аптеками;

– оформление документов инвентаризации лекарственных средств;

– оформление приходных и расходных кассовых ордеров непосредственно в аптеках;

– оформление документов перемещения денежных средств между аптеками;

– оформление чеков продажи, и по окончании смены сводного отчета по контрольно-кассовой машине, с учетом возвращенных товаров в смену;

– поддержка торгового оборудования: фискальные регистраторы, терминалы сбора данных, сканеры штрих кодов, весовое оборудование, дисплеи покупателя, платежные терминалы, ридеры магнитных карт.

Основным преимуществом этой программы считается ее многофункциональность, т.е. ведение учета большого количества различных документов и простота. К недостаткам необходимо отнести ее высокую стоимость и сложность в первоначальной настройке.

3) Программный комплекс «М-АПТЕКА плюс» предназначен для автоматизации аптек и аптечных сетей. Являясь уникальным продуктом нового поколения, «М-АПТЕКА плюс» существенно снижает трудозатраты и облегчает работу всех сотрудников аптеки, позволяя осуществлять ведение полного цикла учета операций на всех стадиях движения товара: от принятия решения о закупке у поставщика до реализации конечному покупателю и последующей отчетности. Система предназначена для сведения воедино различных элементов деятельности по управлению аптечными предприятиями любых размеров и разных схем работы[7].

Работа программного комплекса «М-АПТЕКА плюс» обеспечивает высокий уровень упорядоченности и формализации технологических

процессов аптечного предприятия, позволяет максимально оптимизировать процесс продажи товаров и управления товарными запасами, а также эффективно осуществлять оперативный и аналитический контроль на аптечном предприятии.

Основные возможности программного продукта:

– большой выбор конфигураций системы: от аптечного пункта до оптового склада;

– модульность системы, позволяющая по мере развития бизнеса расширять базовую конфигурацию и экономить средства;

– модули, раздвигающие привычные границы аптечного бизнеса (on-line платежи, запрос наличия товара в сети и многие другие);

– интегрированные в систему электронные справочники лекарственных средств;

– расширенные возможности работы с поставщиками, работа по сводным прайс-листам;

– увеличение оборота и ассортимента товаров;

– возможность автоматического формирования заказов;

– возможность выбора ценообразования;

– развитая дисконтная система (накопительная, пенсионная, скидка для выходного дня и т.д.);

– взаимодействие с бухгалтерскими системами;

– организация работы аптечных сетей, управление ассортиментом в точках продаж.

Недостатками данного продукта являются высокая стоимость, а также большие требования к аппаратному обеспечению.

На основании проведенного анализа, необходимо разработать такой программный продукт, который отвечал бы всем требованиям учета и продажи аптечных товаров, а также был в свободном доступе, и имел минимальную нагрузку на аппаратную часть современных компьютеров.

1.3 Обоснование необходимости разработки автоматизированной подсистемы внутри организации

В настоящее время коммерческим предприятием приходится функционировать в сложных финансово-экономических условиях. Залогом успешной деятельности предприятия в таких условиях является максимально эффективная организация деятельности предприятия на всех уровнях. Это сложная задача, успешное решение которой находится не в области какой-либо одной науки, а обычно на пересечении многих дисциплин, таких как управление и менеджмент, логистика, бухгалтерский учет и, конечно же, информатика.

Автоматизация бизнес процессов современного предприятия является необходимым и обязательным условием его успешного функционирования. Трудно представить сегодня предприятие, на котором, ведется вручную, например, бухгалтерский учет. Но бухгалтерский учет - далеко не единственный пример приложения автоматизации на предприятии. Существующие информационные технологии позволяют автоматизировать деятельности практически всех уровней предприятия. Исходя из этого, становится ясно, что автоматизация деятельности становится ключевой задачей, которую необходимо решить руководству для того, чтобы предприятие работало максимально эффективно.

В данной работе рассматривается автоматизация учета предприятия, которое занимается продажей медицинских товаров, что, естественно, накладывает свою специфику на задачу автоматизации. Далее рассмотрели процесс деятельности предприятия, с использованием функциональных моделей методологии IDEF0. Методология функционального моделирования IDEF0 основана на описания системы в целом как множества взаимозависимых действий или функций. В основу проектирования функциональных моделей системы для учета закупки и продажи товаров был рассмотрен структурный подход. Сущность структурного подхода к

разработке программного обеспечения заключается в его декомпозиции на автоматизируемые функции: система разбивается на функциональные подсистемы, которые, в свою очередь, делятся на подфункции и так далее до конкретных процедур. При этом система сохраняет целостное представление, в котором все составляющие компоненты взаимосвязаны. На рисунке 1.2 представлена деятельность предприятия. Входными ресурсами являются: товар, информация о товаре, информация о поставщиках и заявка покупателя. Выходные ресурсы: проданный товар, информация о продаже, возвращенный товар, информация о списании и отчеты. Механизмом управления является персонал, а правилами управления являются: устав и действующие законодательство.

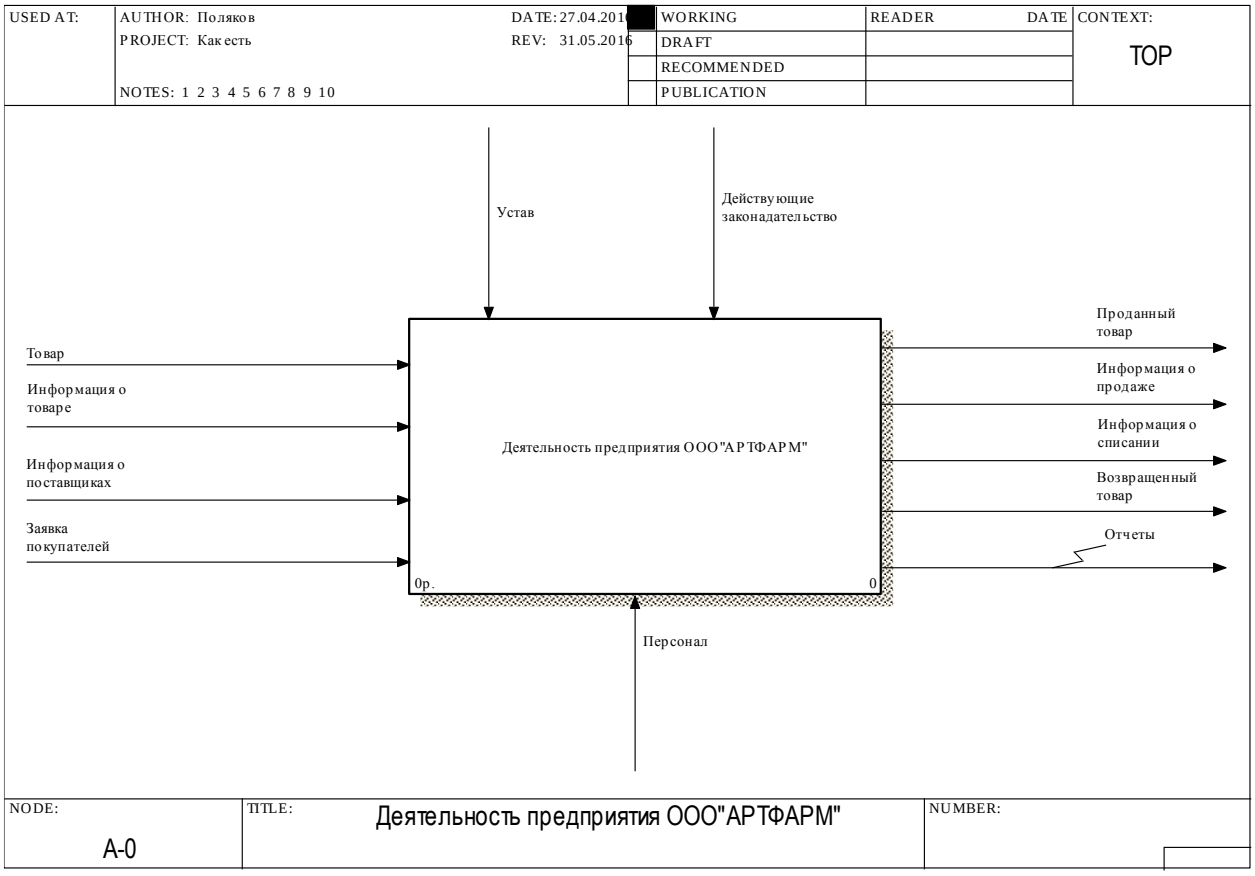

Рисунок 1.2 - Контекстная диаграмма «Деятельность предприятия ООО «АРТФАРМ»

Декомпозировали контекстную диаграмму «Деятельность предприятия ООО «АРТФАРМ» (см. рисунок 1.3):

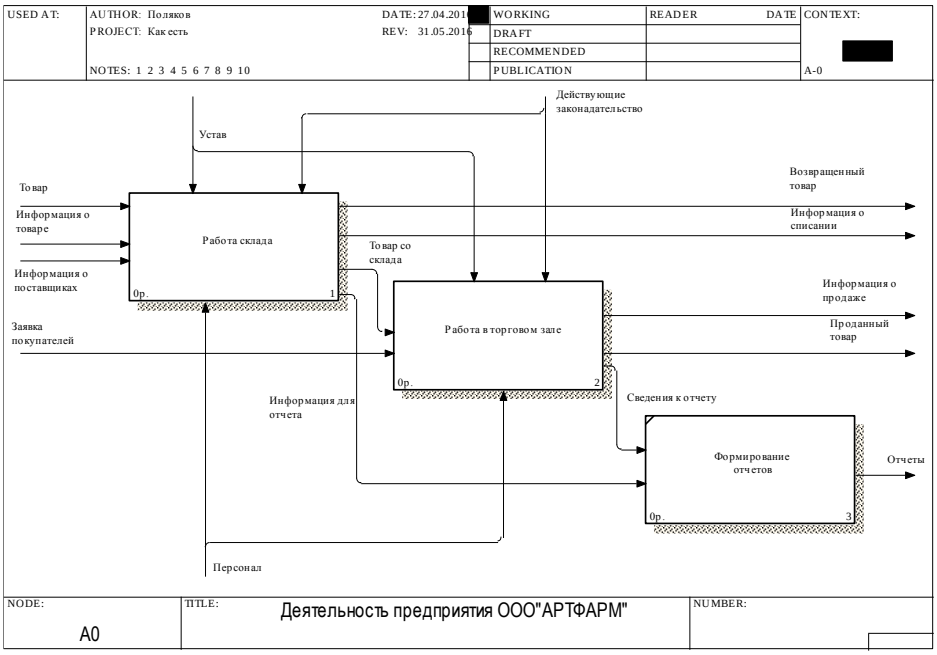

Рисунок 1.3 - Диаграмма декомпозиции контекстной диаграммы

Далее декомпозировали функциональный блок «Работа склада» на 4 действия (см. рисунок 1.4):

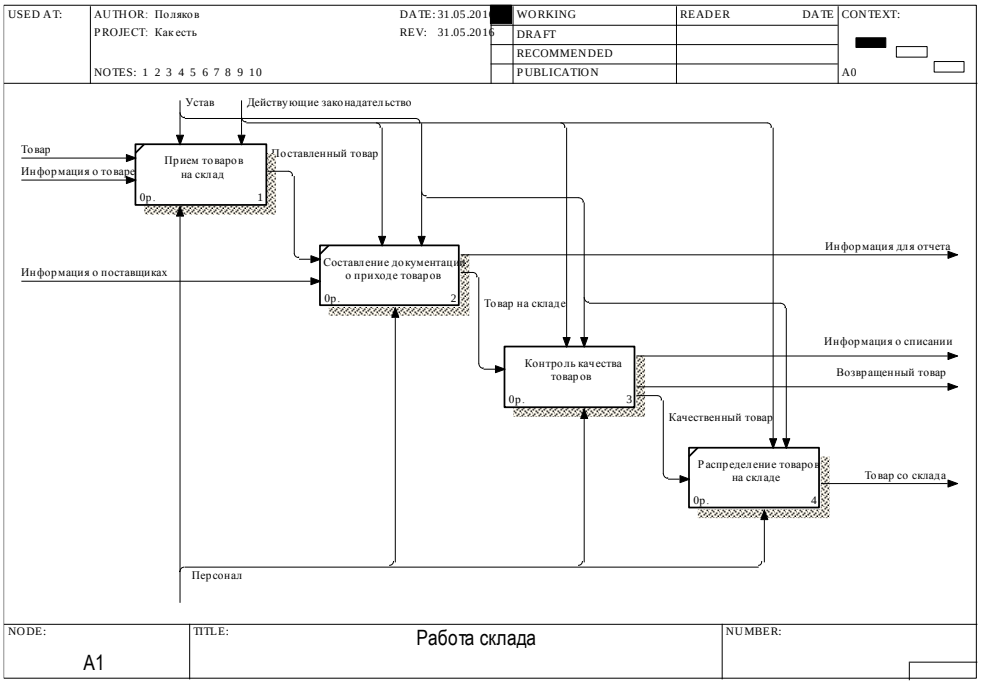

Рисунок 1.4 - Декомпозиция блока «Работа склада»

Декомпозировали функциональный блок «Работа в торговом зале» (см. рисунок 1.5).

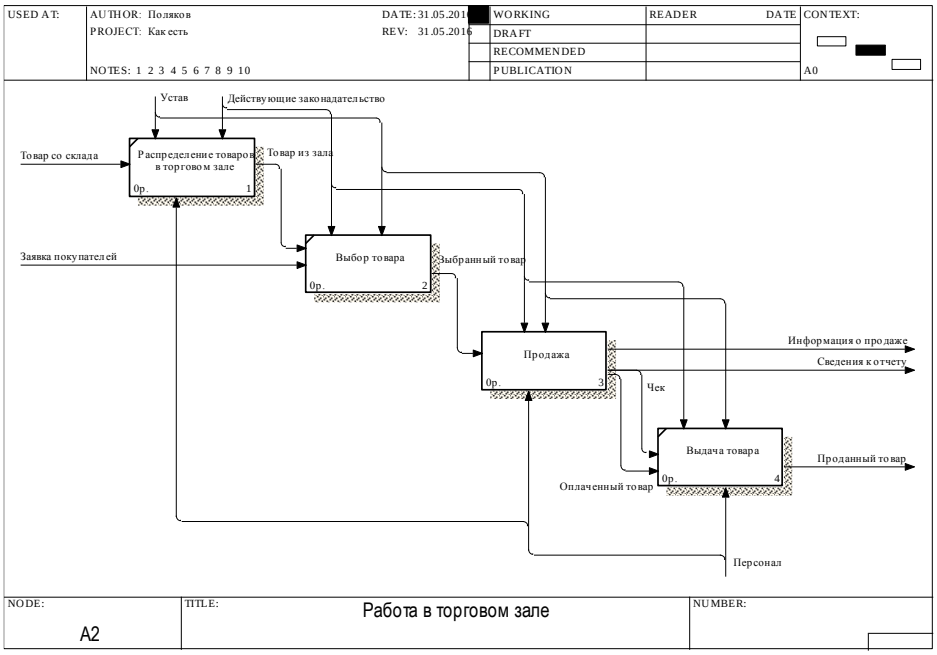

Рисунок 1.5 - Декомпозиция блока «Работа в торговом зале»

Декомпозировали функциональный блок «Контроль качества товаров» (см. рисунок 1.6).

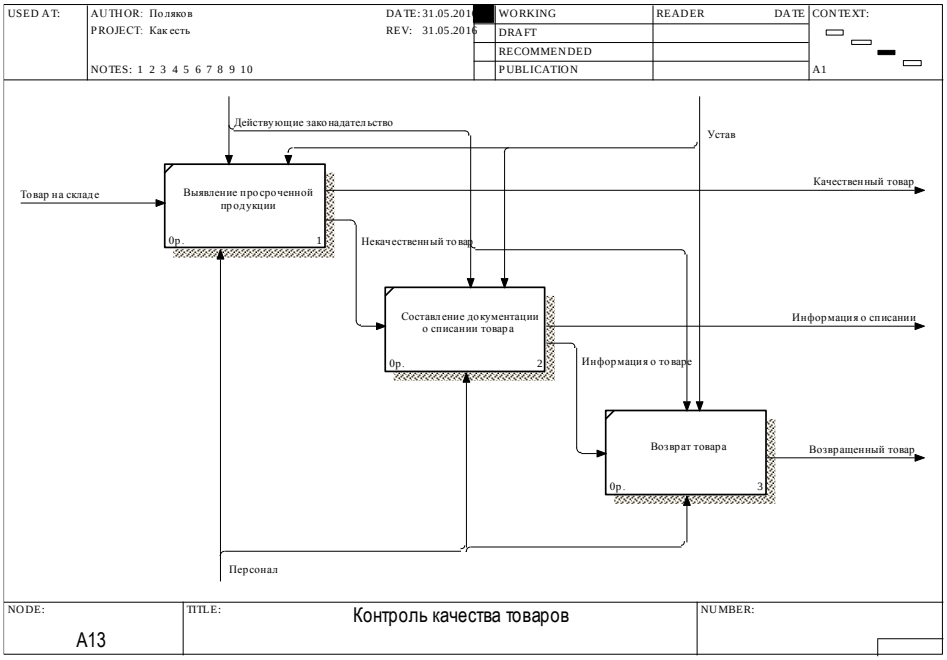

Рисунок 1.6 - Декомпозиция блока «Контроль качества товаров»

Декомпозировали функциональный блок «Продажа» (см. рисунок 1.7)

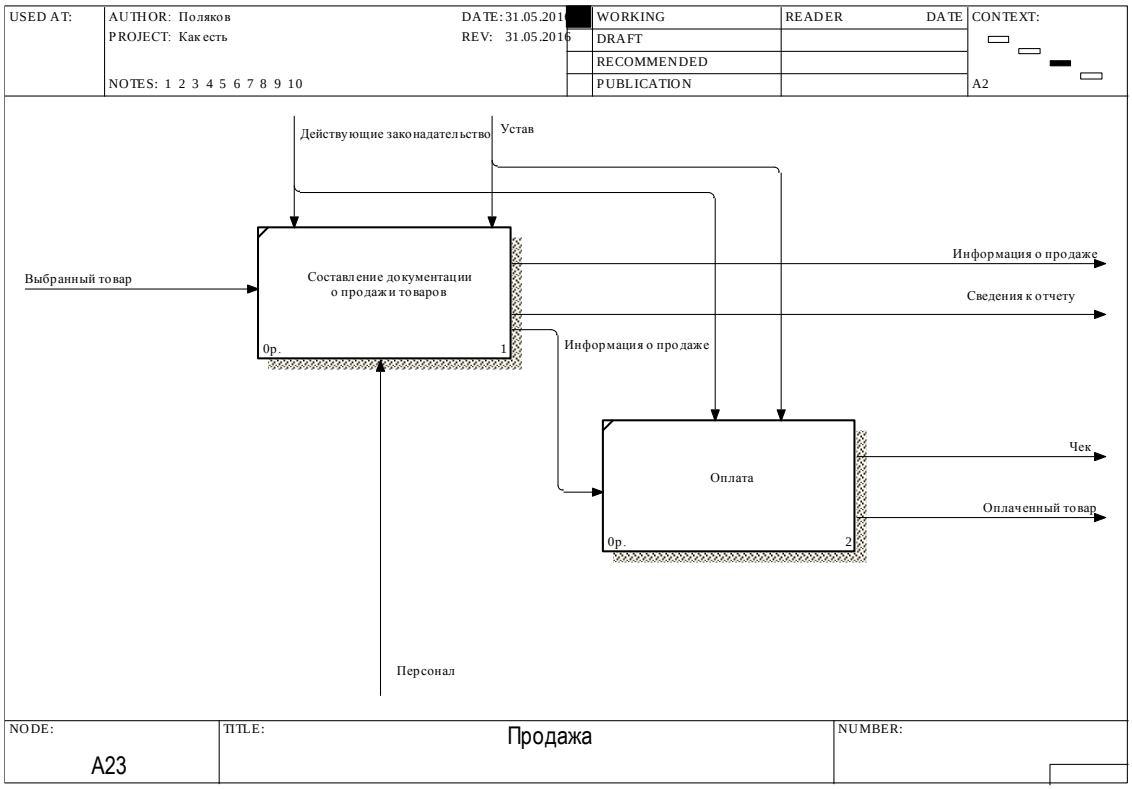

Рисунок 1.7 - Декомпозиция блока «Продажа»

Анализ деятельности предприятия приводит к выводу, что ведение учета традиционными средствами с использованием бумажных документов возможно, но неэффективно. Прежде всего, такой вывод следует из анализа количества документов при ведении учета и продажи товаров. Для большой сети аптек количество таких документов может достигать такого количества, что ручное их оформление становится просто невозможным. Также очень тяжело хранить и осуществлять поиск информации о товарах и их свойствах, документов покупки и продажи в бумажном виде. Намного удобнее и быстрее искать эту информацию в информационной системе, чем в папках бумажных документов.

Однако использование автоматизированной подсистемы дает не только удобство и быстроту поиска информации и оформления документов, но и поднимает эффективность работы на принципиально новый уровень,

предоставляя функции, ранее недоступные. Прежде всего, это касается подсистемы аналитической информации. При использовании системы бумажного учета получить информацию о совершенных сделках, проанализировать наиболее востребованные товары можно, только подняв всю документацию и договора. Благодаря автоматизированной подсистеме это станет гораздо быстрее и доступнее, что позволит лучше оценивать востребованность разных товаров на рынке.

Еще одним неоспоримым преимуществом использования автоматизированной информационной подсистемы является безопасность хранения информации. При хранении информации на бумажных носителях потеря любого бумажного документа является невосполнимой. Такая потеря может произойти как в результате действий злоумышленников, так и в результате действия случайных факторов. При хранении информации в электронном виде существуют методы обеспечения ее безопасного хранения, а также дублирование этой информации, что делает ее еще более надежной.

Таким образом, можно выделить следующие основные преимущества использования автоматизированной информационной системы на основе вычислительной техники для решения задачи построения подсистемы учета:

– повышение удобства поиска и отбора данных из справочников хранения статической информации и журналов выполненных операций;

– повышение скорости поиска и отбора информации, а также оформления операций с покупкой и продажей товаров;

– обеспечение безопасности хранения информации;

– обеспечение многопользовательской работы.

2 ОБОСНОВАНИЕ ПРОЕКТНЫХ РЕШЕНИЙ АВТОМАТИЗИРОВАННОЙ ПОДСИСТЕМЫ УЧЕТА И ПРОДАЖИ МЕДИЦИНСКИХ ТОВАРОВ

2.1 Техническое обеспечение рассматриваемой подсистемы

Техническое обеспечение (ТО) – это комплекс технических средств, предназначенных для работы автоматизированной информационной системы, а также соответствующая документация на эти средства и технологические процессы[10].

Различают следующие виды организации технических средств: централизованная, частично или полностью децентрализованная.

Централизованное техническое обеспечение базируется на применении в автоматизированных информационных системах (АИС) больших электронно-вычислительных машин (ЭВМ) и вычислительных центрах.

Децентрализация технических средств предполагает реализацию функциональных подсистем на персональных компьютерах непосредственно на рабочих местах.

Наибольшее распространение получает частично децентрализованный подход - это организация технического обеспечения на базе распределённых сетей, состоящих из персональных компьютеров и одной или нескольких больших ЭВМ для хранения баз данных, общих для любых функциональных подсистем.

Так как в данной работе организация является сетью аптек, то используется частично децентрализованный подход организации технического обеспечения. Этот подход реализуется на основе архитектуры клиент-сервер. Эта архитектура представляет собой сетевое окружение, в котором управление данными осуществляется на серверном узле, а другим узлам предоставляется доступ к данным. А это означает, что программа

представления данных находится на машине пользователя (на клиенте), а программа управления данными и сами данные на сервере. Клиент, как правило, представляет собой персональный компьютер или рабочую станцию, запрашивающую информацию у сервера. Сервер это компьютер, хранящий информацию, с которой работают клиенты.

Автоматизированная подсистема учета и продажи медицинских товаров предназначена для работы в операционной системе Windows. Для эффективной работы разрабатываемой подсистемы необходимо обеспечить такие требования к архитектуре и параметрам технических средств, которые позволили бы эффективно функционировать самой операционной системе.

Во-первых, должно быть установлено соединение между сервером и клиентом.

Во-вторых, типовая конфигурация клиента должна иметь характеристики, представленные в таблице 1.

Таблица 1 - Типовая конфигурация клиента.

Корпус	Настольный
Процессор	Intel и совместимые системы (1000 ГГц и выше)
Оперативная память	2GB DDR2
Жесткий диск	500GB SATA hard drive
Видеокарта	ASUS Radeon R250 1GB.
Звуковая карта	Интегрированный восьмиканальный звук
Сетевая карта	Интегрированная Gigabit Ethernet 10/100/1000Base-TX
Клавиатура	Keyboard logitech PS/2
Мышь	Оптическая 2-кнопочная с колесом, PS/2
Монитор	ViewSonic vs10781, 19 дюймов

Также клиент должен иметь установленный сетевой протокол TCP/IP, для подключения к сети. Для печати отчетов и выходных документов

необходим принтер, совместимый с компьютером вышеперечисленной комплектации.

В третьих, типовая конфигурация сервера должна иметь характеристики, представленные в таблице 2.

Таблица 2 - Типовая конфигурация сервера.

Процессор	Intel и совместимые системы (двухпроцессорные и более 2000 ГГц и выше)
Видеоконтроллер	Интегрированный Matrox G200eW
Сетевой адаптер	Интегрированный 2-канальный Intel PRO/1000 T Server Adapter (i82574L), 10/100/1000 Mbit/s
SATA-контроллер	Интегрированный, 6 портов SATA 3 Gbps
Оперативная память	RAM 2 GB DDR2-667
Жесткие диски	HDD SATA 500GB
Операционная система	Microsoft Windows Server 2008 R2 Standard x64 Edition RUS
Корпус	Настольный
Источники бесперебойного питания	865W, фиксированный
Клавиатура	Keyboard Logitech, PS/2
Мышь	Оптическая, 2-кнопочная с колесом, PS/2

В-четвертых, должны быть созданы нормальные климатические условия, при которых должны обеспечиваться заданные характеристики, предъявляемым к техническим средствам в части условий их эксплуатации.

2.2 Информационное обеспечение рассматриваемой подсистемы

Назначение подсистемы информационного обеспечения состоит в своевременном формировании и выдаче достоверной информации для принятия управленческих решений.

Информационное обеспечение (ИО) - это совокупность единой системы классификации и кодирования информации, унифицированных систем документации, схем информационных потоков, циркулирующих в организации, а также методология построения баз данных[12]. Структурно информационное обеспечение АИС состоит из двух частей: внемашинного информационного обеспечения (система классификаций и кодирования, система документирования, система документооборота информационных потоков) и внутримашинного информационного обеспечения (информационный фонд).

В данной работе информационное обеспечение будет представлено в виде массивов информации. Массивы информации – это данные по предметной области, организованные в виде базы данных. Под данными понимают, информацию, пригодную для ее передачи и обработки специальными автоматизированными средствами. База данных это совокупность этих данных, систематизированных таким образом, чтобы они были найдены и обработаны. Бывают следующие модели баз данных:

- иерархическая;
- объектно-ориентированная;
- объектно-реляционная;
- реляционная
- сетевая
- функциональная;

Эти модели отличаются не только способом физического управления хранением и поиском данных, но также концептуальными моделями, которые они предоставляют пользователю и программисту. В этой работе

используется реляционная модель баз данных. Тот факт, что в последние годы реляционная модель стала признанным стандартом разработки базы данных, объясняется как мощью самой реляционной модели, так и тем, что она поддерживает стандартный интерфейс SQL, который позволяет различным инструментальным средствам и программным продуктам работать с данными согласованным и понятным способом. Реляционная база данных, представляет собой множество взаимосвязанных таблиц, каждая из которых содержит определенный набор данных. Каждая строка таблицы содержит сведения об одном объекте, а столбы содержат различные характеристики этих объектов или по-другому атрибуты объекта.

При проектировании базы данных необходимо определить все объекты и их свойства, которые необходимо задать в базе данных. Этот процесс называется моделированием данных. Моделью данных является логическое представление структуры данных, которые используется при создании базы данных. Модель, в которой участвуют объекты, а не таблицы, создаваемые в дальнейшем на основе этих объектов, называется концептуальной моделью данных.

Можно представить два различных подхода к проектированию базы данных с помощью концептуальных моделей: проектирование прикладной базы данных снизу вверх и разработка проблемно-ориентированной базы данных сверху вниз. Эти подходы, рассмотренные ниже, позволяют создать базы данных различной структуры.

Модели данных позволяют представить информацию определенным образом. При этом можно разрабатывать базу данных снизу вверх и начинать с просмотра данных на дисплее и (или) в напечатанном виде. Такой подход применяется для создания прикладной базы данных.

Если нужно построить простую базу данных для работы с объектами одного типа, то проектирование снизу вверх может служить хорошую службу, т.к. способы представления и свойства включенных объектов обычно хорошо известны. Однако недостатком такого подхода является то,

что он часто приводит к созданию нескольких баз данных, содержащих повторяющую информацию.

Проектирование сверху вниз опирается на проблемно-ориентированные базы данных. Базы данных, содержащие таблицы, связанные с одним классом предметов или функций, называются проблемно-ориентированные базы данных. В этом случае ход разработки определяют свойства объектов, а не использующие их процедуры.

Проектирование проблемно-ориентированных баз данных включает разработку схем используемых объектов и выявление связей между ними, а также создание моделей всех задействованных баз данных.

Затем нарисованную структуру базы данных отдают на ознакомление пользователям, и опрашивают их, чтобы определить накладываемые ими требования на информацию, содержащуюся в базе данных. Сравнительный анализ обоих подходов показывает, что было бы лучше проектировать базу данных путем выявления групп объектов, с помощью которых можно решить поставленную задачу.

На рисунке 2.1 показана модель данных разрабатываемой подсистемы.

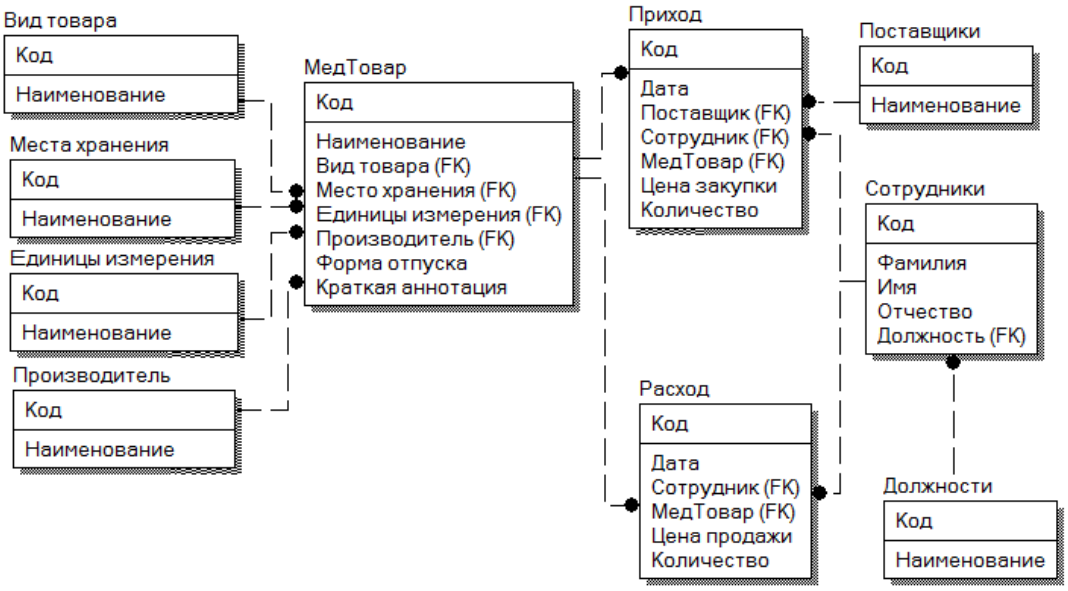

Рисунок 2.1 - Модель данных

Основные объекты системы:

Вид товара - объект, справочный элемент системы, предназначенный для хранения данных о видах товаров;

Места хранения - объект, справочный элемент системы, соответствующий месту хранения товаров;

Единицы измерения - объект, справочный элемент системы, предназначенный для хранения данных о единицах измерения;

Поставщик - объект, соответствующий реальному поставщику.

Сотрудники - объект, соответствующий реальному лицу предприятия.

Должности - объект, справочный элемент системы, предназначенный для хранения данных о должностях предприятия;

Производитель - объект, соответствующий реальному изготовителю.

МедТовар - объект, предназначенный для хранения данных о номенклатурах.

Приход - объект, предназначенный для хранения данных обо всех пришедших номенклатурах.

Расход - объект, предназначенный для хранения данных обо всех расходных операциях аптеки.

Как видно из рисунка все таблицы связаны между собой отношением «один-ко-многим». Это отношение связывает одну строку первой таблицы с несколькими записями второй с помощью первичного или уникального ключа базовой таблицы и соответствующего ему внешнего ключа связанной таблицы. Также видно, что все связи являются не идентифицирующей, так как ни один из первичных ключей дочерней таблицы не содержит внешний ключ родительской таблицы. Каждую запись таблицы можно однозначно идентифицировать то первичному ключу таблицы.

На основе построенной модели данных строится даталогическая модель данных. Даталогическая модель отражает логические связи между элементами данных вне зависимости от их содержания и среды хранения. По сути даталогическая модель отражает существующую схему взаимосвязи таблиц на языке определенной СУБД (Система управления базами данных). На рисунке 2.2 представлена даталогическая модель разрабатываемой подсистемы.

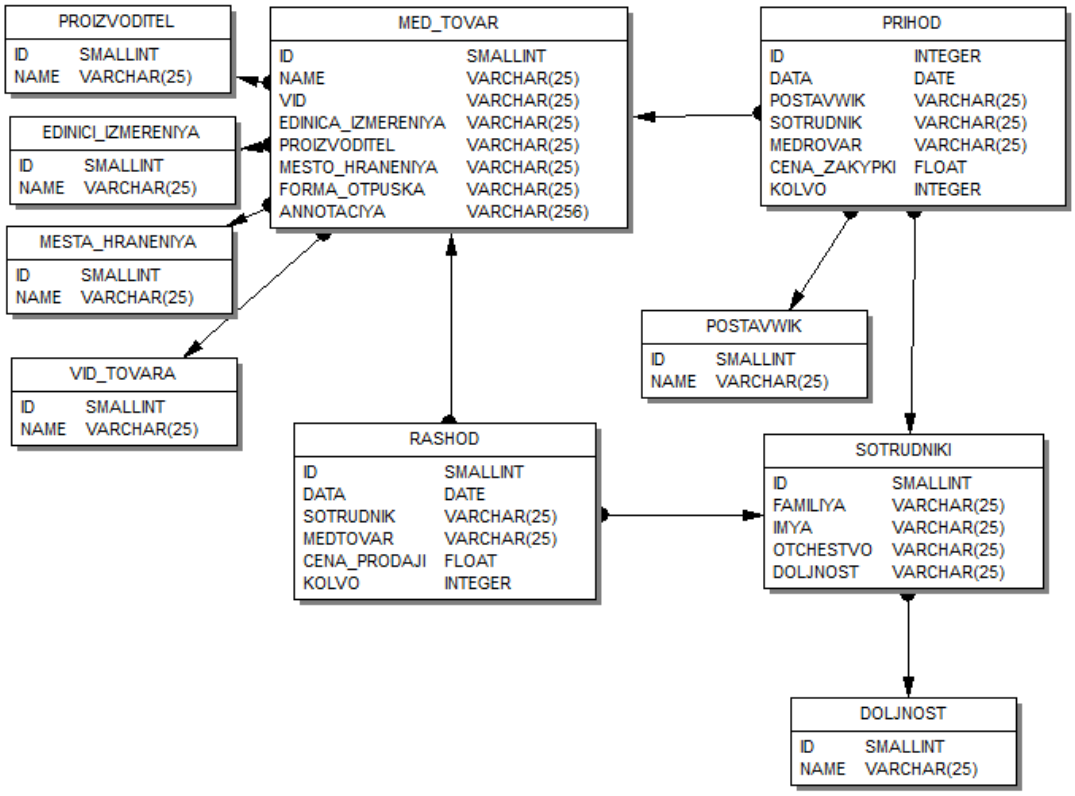

Рисунок 2.2 - Даталогическая модель БД

В таблице 3 приведено более детальное описание таблиц подсистемы.

Таблица 3 – Детальное описание таблиц.

Сущность	Идентификатор таблицы	Атрибут	Идентификатор поля	Тип поля
1	2	3	4	5
Единицы измерения	EDINICI_IZMERENIYA	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
Вид товара	VID_TOVARA	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
Места хранения	MESTA_HRANENIYA	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
Производители	PROIZVODITEL	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
Должности	DOLJNOST	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
Поставщики	POSTAVWIK	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR

Продолжение таблицы 3.

1	2	3	4	5
Сотрудники	SOTRUDNIKI	Код	ID	SMALLINT
		Фамилия	FAMILIYA	VARCHAR
		Имя	IMYA	VARCHAR
		Отчество	OTCHESTVO	VARCHAR
		Должность	DOLJNOST	VARCHAR
Мед. Товар	MED_TOVAR	Код	ID	SMALLINT
		Наименование	NAME	VARCHAR
		Вид	VID	VARCHAR
		Ед. измерения	EDINICA_IZMERENIYA	VARCHAR
		Производитель	PROIZVODITEL	VARCHAR
		Место хранения	MESTO_HRANENIYA	VARCHAR
		Форма отпуска	FORMA_OTPUSKA	VARCHAR
		Краткая аннотация	ANNOTACIYA	VARCHAR
Расход	RASHOD	Код	ID	SMALLINT
		Дата	DATA	DATE
		Сотрудник	SOTRUDNIK	VARCHAR
		Мед. Товар	MEDTOVAR	VARCHAR
		Цена продажи	CENA_PRODAJI	FLOAT
		Количество	KOLVO	INTEGER
Приход	PRIHOD	Код	ID	SMALLINT
		Дата	DATA	DATE
		Сотрудник	SOTRUDNIK	VARCHAR

		Поставщик	POSTAVWIK	VARCHAR
		Мед. Товар	MEDTOVAR	VARCHAR
		Цена покупки	CENA_ZAKYPKI	FLOAT
		Количество	KOLVO	INTEGER

2.3 Программное обеспечение рассматриваемой подсистемы

Совокупность программ и сопровождающей их документации, используемой при эксплуатации этих программ, называется программным обеспечением (ПО)[13].

Программное обеспечение, можно условно разделить на три категории:

– системное ПО, комплекс программных средств, обеспечивающих работоспособность компьютера или сети.

– прикладное ПО, организует процесс обработки информации на компьютере и обеспечивает нормальную рабочую среду для прикладных программ.

– инструментальное ПО (системы программирования), обеспечивающее разработку новых программ для компьютера на языке программирования.

Исходя, из представленной классификации для решения задач по разработке подсистемы можно выделить следующие программное обеспечение:

Системное ПО:

– операционная система (ОС) - это совокупность программ, управляющая аппаратной частью компьютера, его ресурсами обеспечивающая запуск и выполнение прикладных программ, автоматизацию процессов ввода/вывода. Была выбрана ОС Windows 7 семейства Microsoft Windows. Основными преимуществами Windows 7 над остальными версиями являются: минимальные системные требования, простая и быстрая установка на компьютер, быстрый поиск информации,

возможность архивации и восстановления файлов пользователя, хорошая надежность и безопасность, встроенный центр поддержки, довольно удобный и понятный интерфейс.

Прикладное ПО:

– СУБД Firebird - распространенная система управления базами данных. Эта программа является кроссплатформенной, и широко применяется в различных сферах промышленности для учета и управления всей информацией. Основными преимуществами Firebird над другими СУБД являются: поддержка работы множества пользователей, множество различных версий, кроссплатформенность, малый размер дистрибутива, быстрая обработка баз данных, надежное хранение информации, бесплатное распространение;

– ibexpert - для создания реляционной базы данных;

– microsoft office power point 2010 - для составления презентаций по выполненной работе, представление будущих разработок;

– microsoft office word 2010 - для составления отчетов, документации;

– архиватор winrar - для создания архивов данных;

– информационно-поисковые системы – yandex.ru.

Инструментальное ПО:

Для разработки подсистемы используется интегрированная среда разработки (от англ. Integrated Development Environment, сокращенно IDE) C++ Builder 6 фирмы Borland. Вообще, IDE – это система программных средств, используемая программистами для быстрой разработки программного обеспечения. Данное программное средство было выбрано исходя из следующих факторов:

– поддержка ОС window 7;

– поддержка языка программирования C++;

– более 100 визуальных компонентов;

– наличие компонентов для работы с firebird;

– интуитивное кодирование;
– визуальное наследование и связывание форм;
– редактор кода с символьными ссылками и историей навигации;
– минимальные нагрузки на аппаратную часть компьютера;
– наличие файла справки на русском языке;
– возможность настройки рабочего пространства;
– наличие системы проверки синтаксиса;
– простой и понятный интерфейс;

На основании изложенных факторов можно сказать, что данное ПО полностью подходит для разработки подсистемы учета и продажи медицинских товаров.

3 ПРОЕКТНАЯ ЧАСТЬ

3.1 Разработка и описание функциональной модели

На основании представленных ранее бизнес моделей и проведенного анализа, можно говорить о необходимости автоматизации учета медицинских товаров внутри рассматриваемой организации.

Задача автоматизации работы предприятия состоит из нескольких основных подзадач, а именно:

– создание базы данных с удобным методом хранения;

– ИС должна предоставлять полные возможности ведения базы товаров.

– отсутствие необходимости обслуживания со стороны специально обученного персонала;

– легкая возможность резервного копирования и восстановления;

– легкая возможность модернизации – простота и открытость ИС является одним из основных требований для страховки компании;

– отсутствие высоких требований к аппаратным компонентам – для отсутствия высоких дополнительных затрат на модернизацию парка ЭВМ компании;

– ввод информации в базу данных должен быть разделен для всех таблиц или логических сущностей базы;

– ввод информации в базу должен иметь удобный интерфейс;

– основным путем получения данных из базы должны являться предварительно сформированные формы и отчеты;

Далее разработали функциональную модель с использованием функциональных моделей методологий IDEF0. На рисунке 3.1 представлена деятельность предприятия.

Входными ресурсами являются:

«Товар» - товары, которые покупает и продает рассматриваемая организация;

«Информация о товаре» - информация, необходимая при создании документов покупки и продажи товаров, а также для ведения учета на складах;

«Информация о поставщиках» - информация, необходимая при создании документов покупки товаров;

«Заявка покупателя» - информация о нужном товаре, чтобы сотрудник мог осуществить продажу товара или, основываясь на этой информации подобрать более подходящий.

Выходные ресурсы:

«Проданный товар» - название и количество единиц товара, реализованных клиентам;

«Информация о продаже» - сведения о совершенных операциях по продажи товаров;

«Возвращенный товар» - товар, который не прошел контроль качества;

«Информация о списании» - документы, созданные в процессе контроля качества товаров;

«Отчеты» - аналитическая информация, представленная в удобном виде.

Механизмом управления является персонал и информационная подсистема, а правилами управления являются: устав и действующие законодательство.

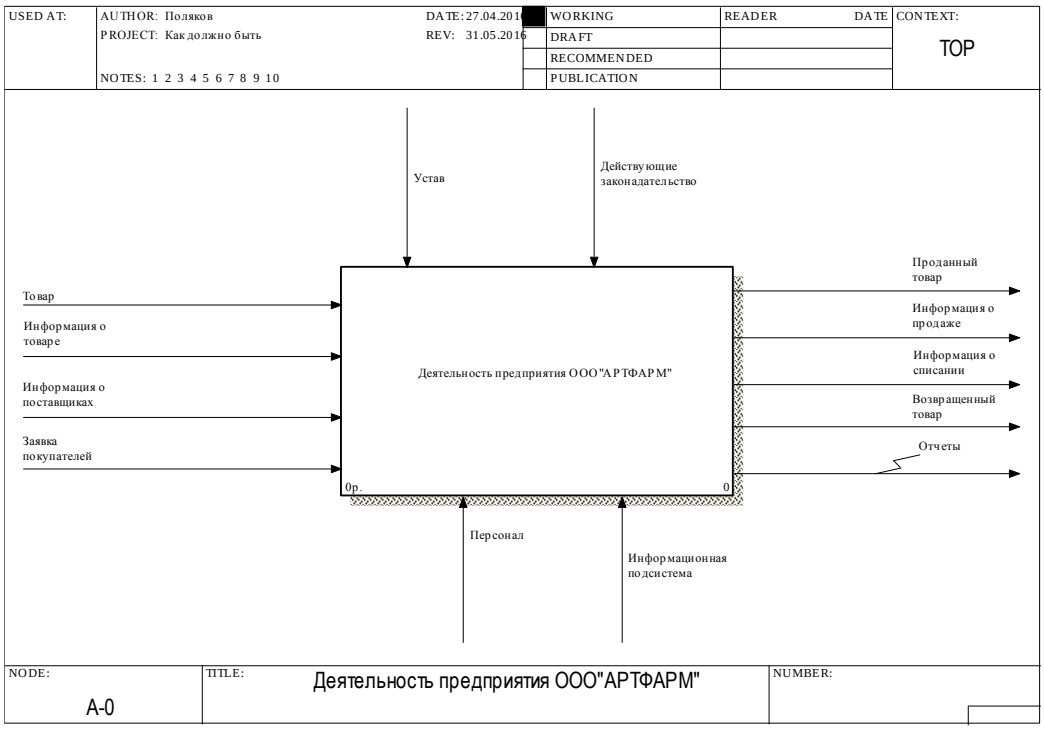

Рисунок 3.1 - «Деятельность предприятия ООО «АРТФАРМ»

Декомпозировали контекстную диаграмму «Деятельность предприятия «ООО АРТФАРМ» (см. рисунок 3.2).

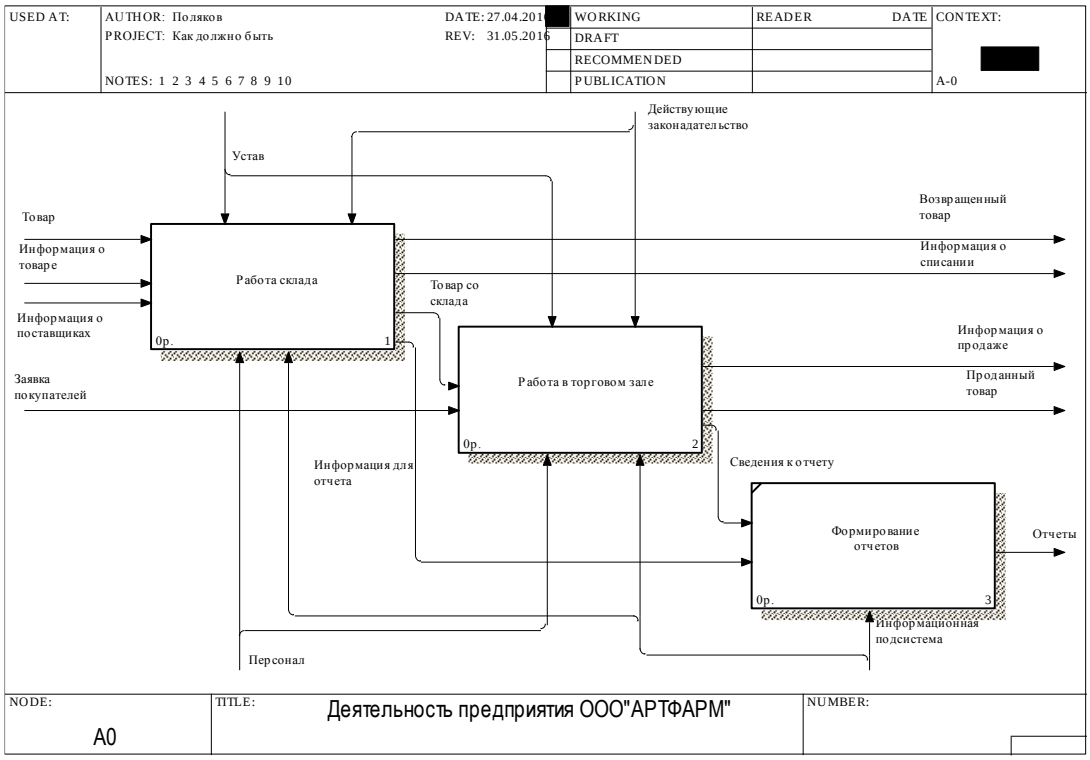

Рисунок 3.2 - Декомпозиция контекстной диаграммы

Декомпозировали функциональный блок «Работа склада» (см рисунок 3.3).

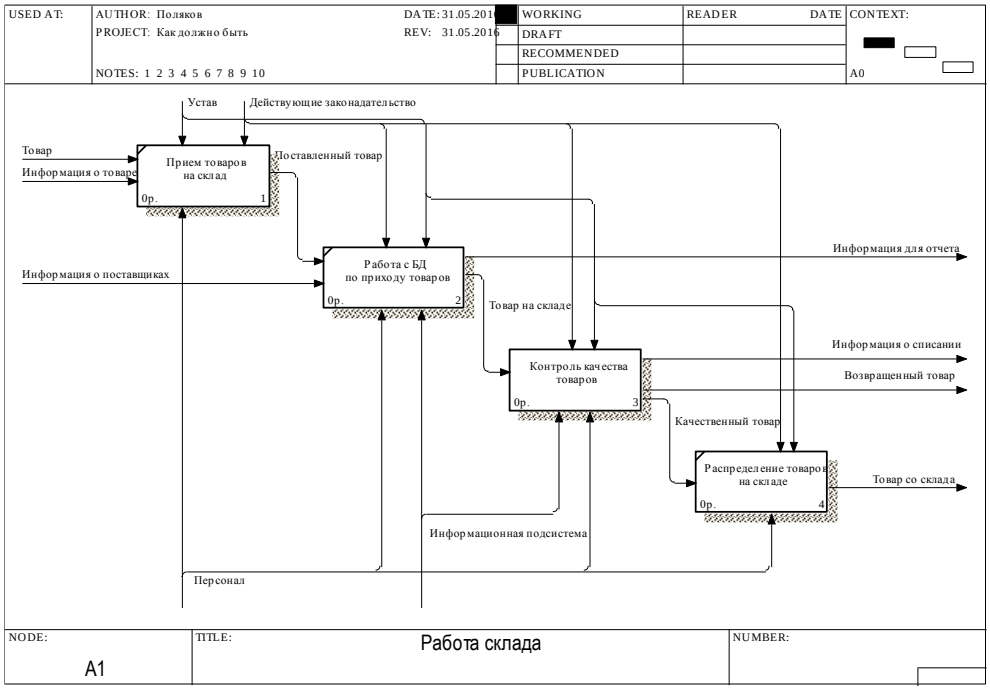

Рисунок 3.3 - Декомпозиция блока «Работа склада»

Декомпозировали функциональный блок «Работа в торговом зале» (см. рисунок 3.4).

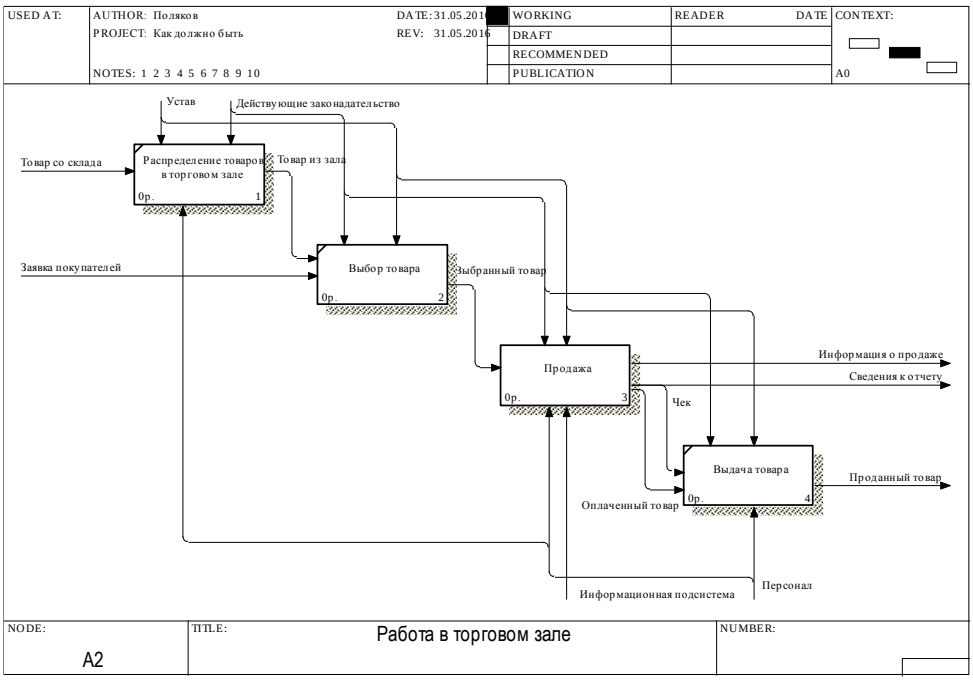

Рисунок 3.4 - Декомпозиция блока «Работа в торговом зале»

Декомпозировали функциональный блок «Продажа» (см. рисунок 3.5).

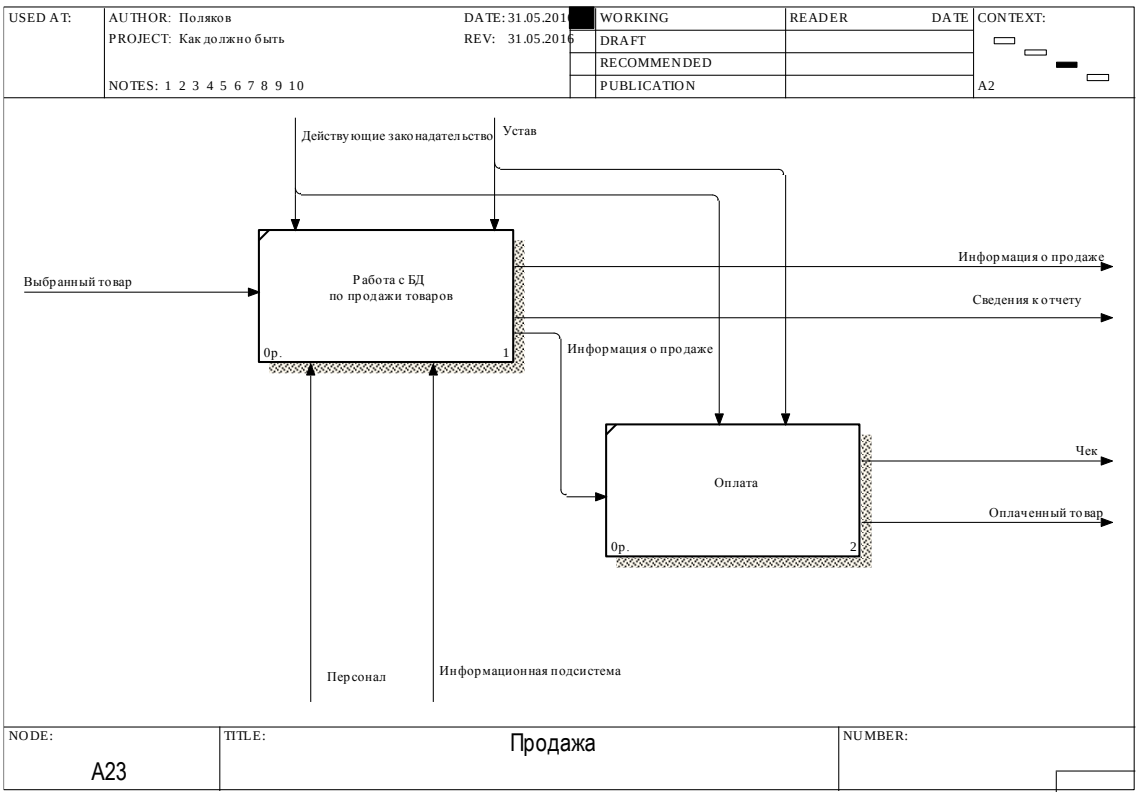

Рисунок 3.5 - Декомпозиция блока «Продажа»

На рисунке 3.3 показан процесс внесения информации о товаре в базу данных. Первоначально товар принимают на склад. Далее вносят информацию по приходу товара в БД. Затем проверяют его на качество. Если товар качественный оставляют его на складе на специально отведенном месте. Если же товар не качественный, то возвращают его на утилизацию. Далее эта информация из БД будет браться при совершении продажи товаров, которое показано на рисунке 3.5. Сначала покупатель сообщает информацию об интересующем его товаре. После, сотрудник использует поиск в базе данных. При наличии данного товара совершается оплата, и выдача соответствующего товара. Все совершенные операции хранятся в информационной подсистеме. Эти данные можно использовать для составления аналитической информации.

3.2 Разработка программных модулей

Разработку автоматизированной подсистемы учета и продажи товаров начинали с создания графического интерфейса пользователя (от англ. graphical user interface, GUI). Графический интерфейс - это неотъемлемая часть программ и информационных систем. Интерфейс пользователя - разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), другая - машиной/устройством. Представляет собой совокупность средств и методов, при помощи которых пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой.

Поскольку интерфейс есть совокупность, то он состоит из элементов, которые, сами по себе, также могут состоять из элементов (так, экран дисплея может содержать в себе другие окна, которые, в свою очередь, могут содержать панели, кнопки и прочие интерфейсные элементы).

Особое и отдельное внимание в интерфейсе пользователя традиционно уделяется его эффективности и удобству пользования. Понятный, удобный, дружественный - его основные характеристики.

В данном разделе приведена только часть программного кода, весь остальной программный код см. приложение А.

Для начала запустили инструментальное средство разработки C++ Builder 6, основное окно которого показано на рисунке 3.6.

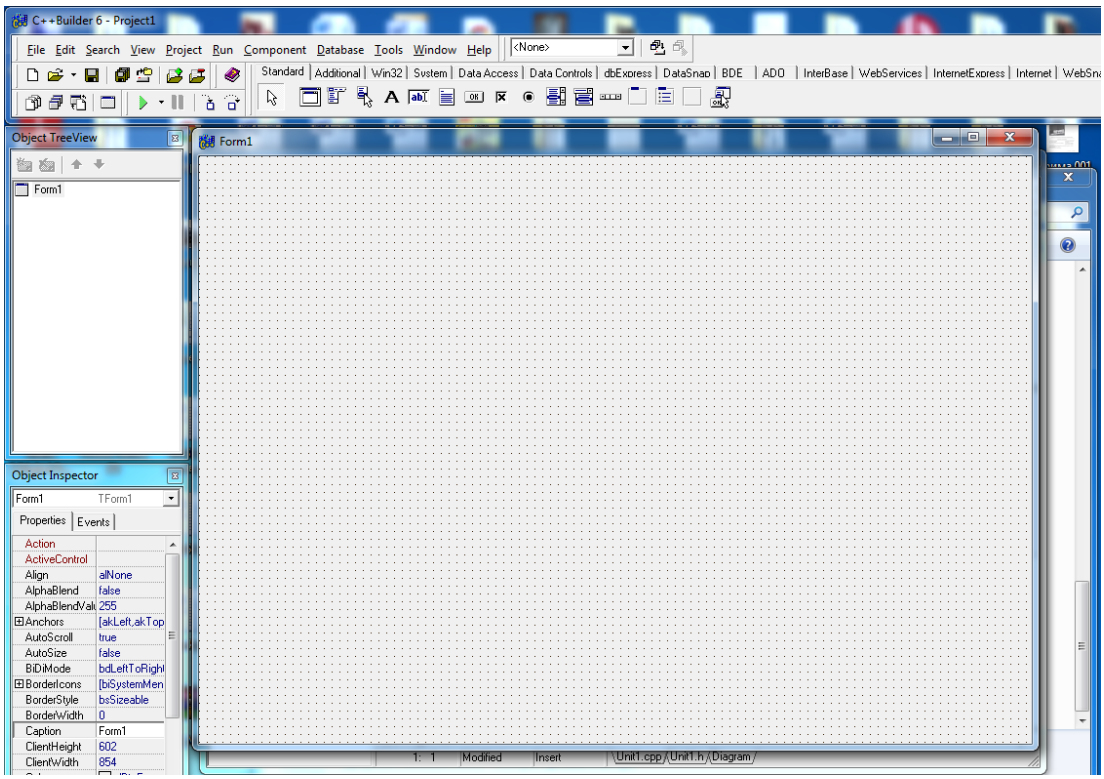

Рисунок 3.6 - Основное окно инструментального средства разработки

Используя главное горизонтальное меню, создали новый проект. Дали ему название «Apteka», и сохранили файлы проекта в каталог на жестком диске. Далее перешли к созданию главного окна приложения. Для этого использовали форму, которая создается автоматически при создании нового проекта. Проделали следующие операции над этой формой:

– дали имя формы – «AptekaMain» не только для более удобного обращения к ней, но и чтобы не возникало путаницы при написании программного кода;

– отключили кнопку управления «Свернуть», и между тем, установили свойство «BorderStyle» на «bsSingle». Теперь растянуть/свернуть данную форму, и впоследствии остальные у пользователя не удастся;

– подобрали подходящие размеры - окно программы не должно занимать большое пространство на дисплее;

– поменяли значение «Caption» на «Подсистема учета и продажи медицинских товаров»;

– установили подходящий стиль и размеры шрифтов. Компоненты, которые будут перенесены на форму, примут такие настройки шрифтов.

Далее перешли к созданию собственного меню. Главное меню программы расположено на левой половине формы. Все кнопки выровнены, и сгруппированы используя компонент типа «Panel» имя которого «MainMenu». Главное меню, представлено следующими кнопками:

– справочники – для отображения всех имеющихся справочников;
– приход товаров – для работы с формой приход товаров;
– расход товаров – для работы с формой расход товаров;
– запросы – для реализации различных запросов;
– отчеты – для создания различных отчетов;
– выручка – для отображения суммарной информации;
– настройки – для создания подключения к БД;
– выход – выход из программы;

Главное окно программы представлено на рисунке 3.7.

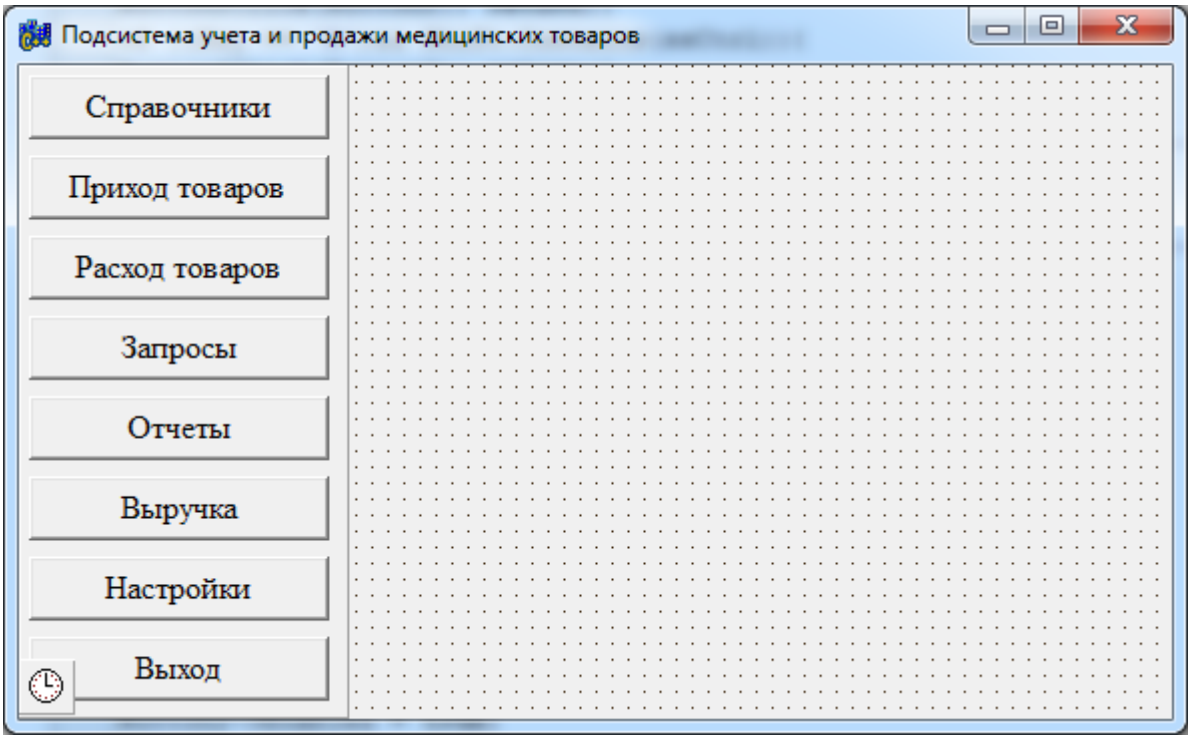

Рисунок 3.7 - Главное окно разрабатываемой подсистемы

Так как в БД имеется не один, а восемь справочников, поэтому аналогичным образом сгруппировали их, используя компонент типа «Panel» имя которого «Spravochniki». Это объединение будет выступать в роли вложенного меню.

Повторили операцию создания вложенного меню, но только для запросов и отчетов.

Далее используя компонент из вкладки «System» - «Timer» реализовали анимированную свертку/развертку вложенного меню. Для этого: установили первоначальные свойства таймера, создали массив указателей на панели вложенного меню, объявили необходимые переменные и в событии «OnTimer» написали программный код, представленный ниже (см. рисунок 3.8).

```
void __fastcall TAptekaMain::Timer1Timer(TObject *Sender)
{
int i;
if (panel[s]->Tag == 2) panel[s]->Tag = 0;

for (i = 0; i < 3; i++) {

if (panel[i]->Tag == 1) {
 if (panel[i]->Left != max_left)
  panel[i]->Left = panel[i]->Left + 15;
 else {Timer1->Enabled = false; panel[i]->Tag = 2; s = i;}
 }
else { if (panel[i]->Tag == 0)  {
        if (panel[i]->Left != 0) panel[i]->Left = panel[i]->Left - 15;
        else { panel[i]->Tag = 0;}
      }
      }
      }
}
```

Рисунок 3.8 - Программный код анимированной свертки/развертки меню

Таким образом, с помощью таймера был создан более привлекательный внешний вид главного окна программы.

Следующий шаг - это создание внешнего вида форм справочников. Каждый справочник отображается в отдельном окне. Для всех форм справочников настроили свойства: «Name», «Caption», «BorderStyle», где значение установлено на «bsToolWindow», и «Position» -

«poMainFormCenter». Отображение информации из таблиц осуществляется с помощью «DBGrid» из вкладки «Data Controls». Дочерние формы будут появляться по центру главной формы. На следующем рисунке представлен внешний вид справочника «Единицы измерения» (см. рисунок 3.9).

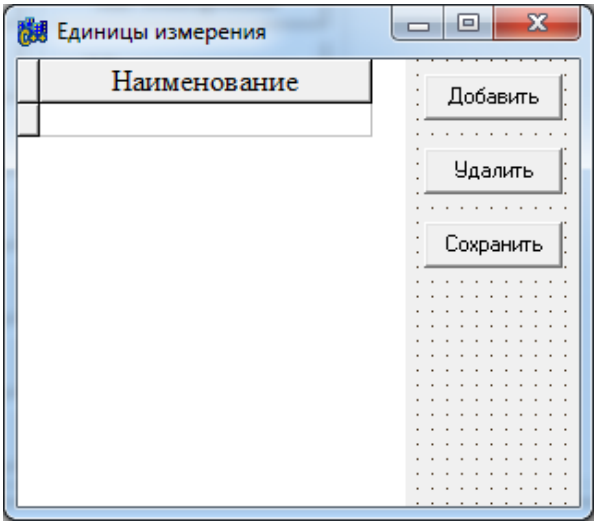

Рисунок 3.9 - Внешний вид справочника «Единицы измерения»

Такие справочники как: виды товаров, единицы измерения, места хранения, производители, поставщики, должности и сотрудники – имеют очень схожий вид. И только лишь справочник «Список медицинских товаров» отличается от них (см. рисунок 3.10).

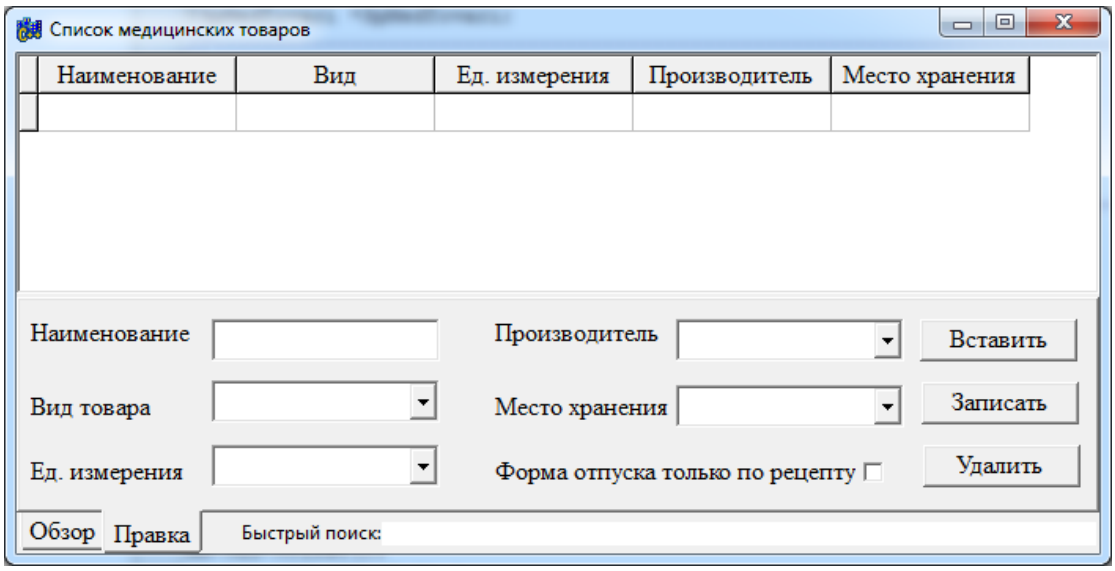

Рисунок 3.10 - Внешний вид справочника «Список медицинских товаров»

Этот справочник, по сути, содержит в себе несколько других. Было создано две вкладки «Обзор» - для отображения и редактирования краткой аннотации о товаре, и «Правка» - для создания и редактирования записи о товаре. В нижней части программы организован быстрый поиск товаров.

Далее реализуем формы документов расхода и прихода товаров. На рисунках 3.11 и 3.12 представлен внешний вид форм документов.

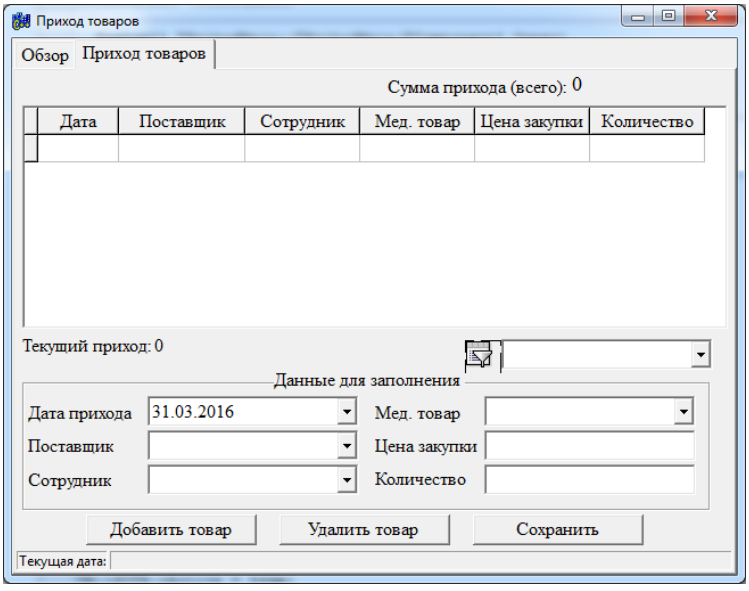

Рисунок 3.11 - Внешний вид документа «Приход товаров»

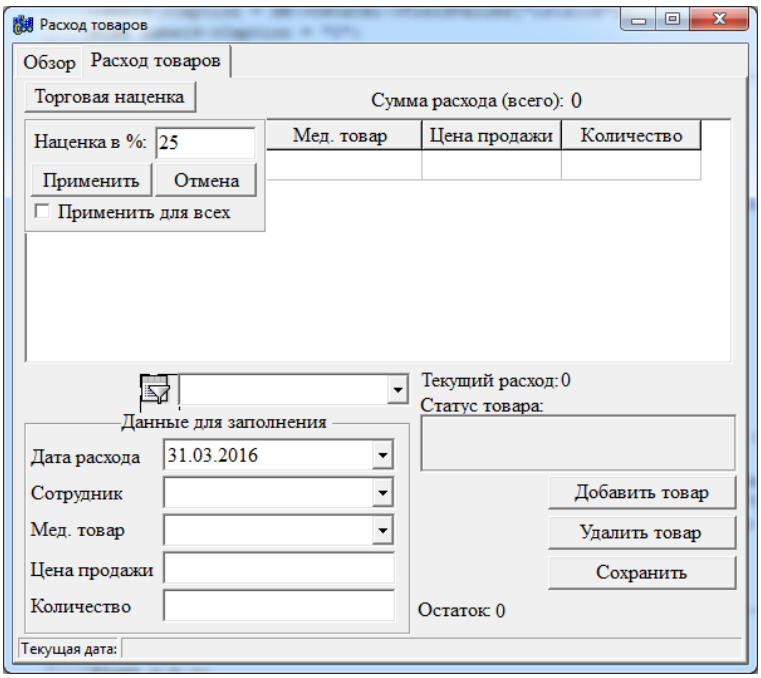

Рисунок 3.12 - Внешний вид документа «Расход товаров»

У обоих документов есть встроенная фильтрация данных, сгруппированные объекты для их заполнения, полоса состояния для отображения системной даты и по три основных кнопок действий. Отличия состоит лишь в том, что в окне расхода товаров есть специальная кнопка для установления наценки на товары, и панель статуса товара.

Завершающим этапом создания графического интерфейса пользователя является создания окна настроек программы. На левой половине формы построены компоненты для подключения к БД, а на правой информация о различных сочетаниях клавиш для быстрого перехода по окнам. При создании использовались различные компоненты, в том числе и «OpenDialog» из вкладки «Dialogs». В данном случае этот компонент предназначен для возвращения строки расположения файла. Внешний вид формы настроек представлен на рисунке 3.13.

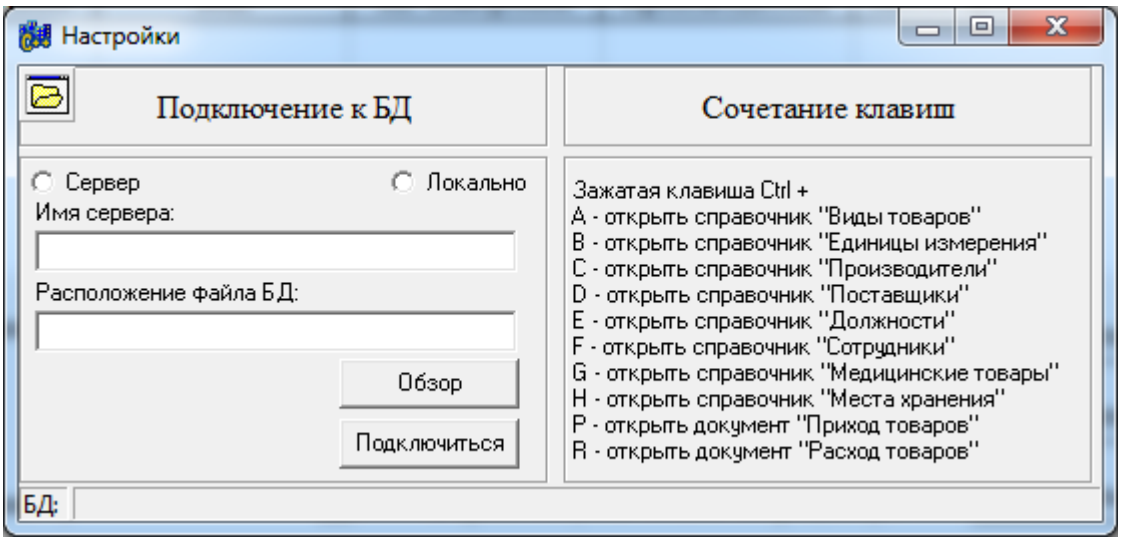

Рисунок 3.13 - Внешний вид окна настроек программы

После создания графического интерфейса пользователя необходимо обеспечить взаимосвязь компонентов. Первоначально установили контакт с базой данных с помощью невизуальных компонентов с вкладки «InterBase». Чтобы не загромождать формы этими компонентами, поместили их в специальный контейнер – DataModule, при этом дав им подходящие имена (см. рисунок 3.14).

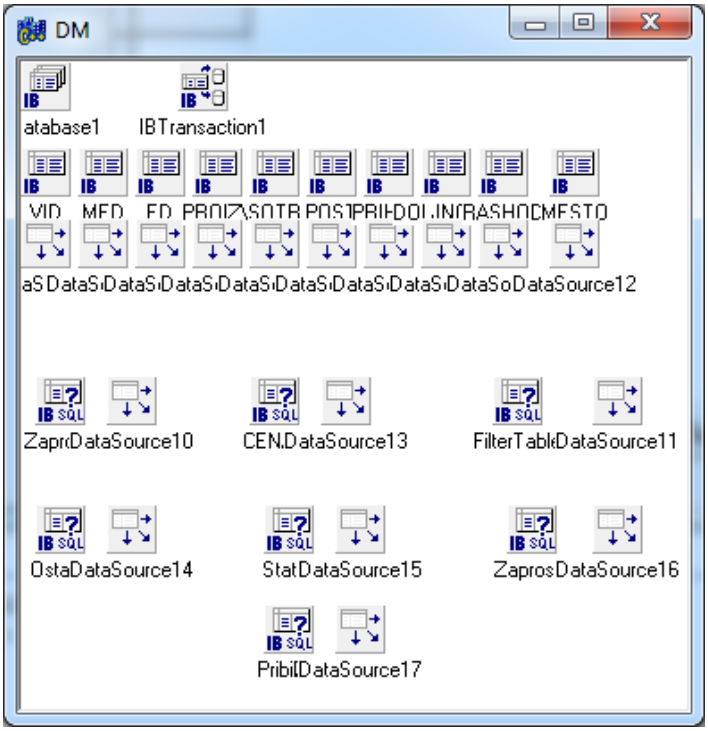

Рисунок 3.14 - Контейнер для невизуальных компонентов

Затем, посредством компонентов «DataSource» обеспечили связь таблиц базы данных и «DBGrid». Первичные ключи таблиц пользователю видеть не обязательно, поэтому сделали их невидимыми.

Для справочников: виды товаров, единицы измерения, места хранения, производители, поставщики, должности и сотрудники, написали программный код для кнопок добавить, удалить и сохранить. Ниже представлен программный код только одного из них – места хранения:

– кнопка добавить (см. рисунок 3.15):

```
void __fastcall TSpMesto::Button1Click(TObject *Sender)
{
DM->MESTO->Insert();
DBGrid1->SetFocus();
}
```

Рисунок 3.15 - Программный код кнопки «Добавить»

– кнопка удалить (см. рисунок 3.16):

```
void __fastcall TSpMesto::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить запись?","Подтвер
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->MESTO->Delete();
}
```

Рисунок 3.16 - Программный код кнопки «Удалить»

– кнопка сохранить (см. рисунок 3.17):

```
void __fastcall TSpMesto::Button3Click(TObject *Sender)
{
if (DM->MESTO->Modified) DM->MESTO->Post();
}
```

Рисунок 3.17 - Программный код кнопки «Сохранить»

Далее перешли к справочнику «Список медицинских товаров». В событие «OnShow» написали следующий программный код (см. рисунок 3.18).

```
void __fastcall TSpMedTovari::FormShow(TObject *Sender)
{
canpost = false;
Edit1->Clear();
CB->Clear();
DM->VID->Active = true;
DM->VID->First();
while (!DM->VID->Eof) {
CB->Items->Add(DM->VIDNAME->AsString);
DM->VID->Next(); }
CB2->Clear();
DM->ED->Active = true;
DM->ED->First();
while (!DM->ED->Eof) {
CB2->Items->Add(DM->EDNAME->AsString);
DM->ED->Next(); }
CB3->Clear();
DM->PROIZV->Active = true;
DM->PROIZV->First();
while (!DM->PROIZV->Eof) {
CB3->Items->Add(DM->PROIZVNAME->AsString);
DM->PROIZV->Next(); }
CB4->Clear();
DM->MESTO->Active = true;
DM->MESTO->First();
while (!DM->MESTO->Eof) {
CB4->Items->Add(DM->MESTONAME->AsString);
DM->MESTO->Next();}
}
```

Рисунок 3.18 - Обработчик события «OnShow»

Дело в том, что для добавления и редактирования товаров пользователю не нужно вбивать информацию вручную, она загружается из других справочников автоматически. Но и также пользователю не удастся изменить эту информацию вследствие того, что был запрещен ввод любых символов (см. рисунок 3.19). Можно лишь осуществить быстрый переход щелкнуть два раза кнопкой мышкой на нужный справочник (см. рисунок 3.20).

```
void __fastcall TSpMedTovari::CBKeyPress(TObject *Sender, char &Key)
{Key = 0;}
```

Рисунок 3.19 - Программный код запрета ввода любых символов

```
void __fastcall TSpMedTovari::CBDblClick(TObject *Sender)
{SpVidTovara->Show();}
```

Рисунок 3.20 - Программный код для быстрого перехода по справочникам

В событие «AfterScroll» таблицы справочника написали следующий программный код (см. рисунок 3.21).

```
void __fastcall TDM::MEDAfterSl(TDataSet *DataSet)
{
SpMedTovari->Edit1->Text = MEDNAME->Value;
SpMedTovari->CB->ItemIndex = SpMedTovari->CB->Items->IndexOf(MEDVID->AsString);
SpMedTovari->CB2->ItemIndex = SpMedTovari->CB2->Items->IndexOf(MEDEDINICA_IZMERENIYA->AsString);
SpMedTovari->CB3->ItemIndex = SpMedTovari->CB3->Items->IndexOf(MEDPROIZVODITEL->AsString);
SpMedTovari->CB4->ItemIndex = SpMedTovari->CB4->Items->IndexOf(MEDMESTO_HRANENIYA->AsString);
if (MEDFORMA_OTPUSKA->Value == "Рецепт") SpMedTovari->CheckBox1->Checked = true;
else  SpMedTovari->CheckBox1->Checked = false;
}
```

Рисунок 3.21 - Обработчик события «AfterScroll»

Это событие наступает после перемещения указателя таблицы на новую запись. Данные выделенной строки передаются в специально размещенные на форме компоненты. Теперь для редактирования записей стоит всего лишь изменить значения этих самых компонентов и нажать кнопку «Записать».

Так как записей о различных медицинских товарах может быть огромное множество, поэтому реализовали быстрый поиск, который расположили отдельно внизу поверх вкладок. Для его реализации использовали метод «Locate». Данный метод возвращает ту строку таблицы, которая удовлетворяет условию. Ниже представлен программный код (см. рисунок 3.22).

```
void __fastcall TSpMedTovari::Edit2Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit2->Text};
DM->MED->Locate("NAME", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
```

Рисунок 3.22 - Программная реализация поиска

Перешли к написанию программного кода для документов. Как и у справочника медицинских товаров, информация из других справочников загружается автоматически. Самим пользователем ввести вручную предстоит только цену на товар и его количество. Цена и количество не могут принимать отрицательные значения, но первое из них может быть, как и целым, так и дробным числом в отличие от второго. Поэтому что бы избежать ошибок при выполнении программы запретили ввод некоторых символов и использовали глобальную переменную DecimalSeparator, которая содержит символ, используемый в качестве разделителя целой и дробной части числа. Ниже представлен программный код (см. рисунок 3.23).

```
void __fastcall TPrihodMain::Edit2KeyPress(TObject *Sender, char &Key)
{
 if ( ( Key >= '0') && ( Key <= '9' ) ) return;
 if ((Key == ',') || (Key == '.'))
    {  Key = DecimalSeparator;
       if ( (Edit2->Text).Pos(DecimalSeparator) != 0 )
           Key = 0;
       return;
    }
 if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
    {
       Edit3->SetFocus();
       return;
    }
    Key = 0;
}
```

Рисунок 3.23 - Программный код для ограничения ввода символов

При работе с документом приходится работать с большим количеством записей, поэтому лучшим решением отбора нужных записей является его преднамеренная фильтрация. В программе фильтр для обоих документов выглядит, как показано на рисунке 3.24.

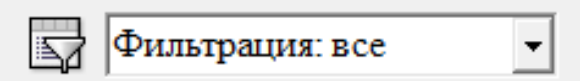

Рисунок 3.24 - Внешний вид участка окна программы для фильтрации данных

Пиктограмма используется для сброса фильтрации. Условиями отбора являются записи справочника «Виды товаров» (см. рисунок 3.25).

```
void __fastcall TPrihodMain::FilterChange(TObject *Sender)
{
DM->FilterTable->Close();
DM->FilterTable->SQL->Clear();
if (Filter->ItemIndex == 0)
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR");
else
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR where VID LIKE '%"+Filter->Text+"'" );
DM->FilterTable->Open();
CB3->Clear();
DM->FilterTable->First();
while (!DM->FilterTable->Eof) {
CB3->Items->Add(DM->FilterTable->FieldByName("NAME")->AsString);
DM->FilterTable->Next();}
}
```

Рисунок 3.25 - Программная реализация фильтрации данных

У справочника «Расход товаров» есть специальная панель, которая отражает статус товара. Эта панель может отражать три возможных значения: «Товар еще не был в продаже», «Товар закончится менее чем через 7 дней. Необходимо сделать заказ» и «Заказ товара не требуется». Это реализуется с помощью двух запросов, один из которых отражает остаток товара на складе, а другой среднее значение продажи товара в день. В процессе работы программы остаток делится на среднее значение, полученный результат сравнивается с цифрой семь и выводится соответствующий результат (см. рисунок 3.26).

```
float ave;
DM->Status->Close();
DM->Status->SQL->Clear();
DM->Status->SQL->Add("select sr from status where medtovar = '" + CB2->Text + "'
DM->Status->Open();
if (DM->Status->FieldByName("SR")->IsNull) Label14->Caption = "Товар не был в пр
else { ave = StrToInt(Label9->Caption) / DM->Status->FieldValues["SR"];
 if (ave < 7) Label14->Caption = "Товар закончится менее чем через 7 дней. Необх
 else Label14->Caption = "Заказ товара не требуется.";
 }
```

Рисунок 3.26 - Фрагмент программного кода для реализации статуса товара

И наконец, перешли к программированию настроек программы. Необходимо учитывать тот факт, что приложение может быть перенесено с одно компьютера на другой большое количество раз. Была обеспечена

пользовательская организация соединения с БД. Теперь пользователь должен сам указать расположения файла базы данных. Если этого не сделать, то при открытии на другом компьютере, приложение выдаст ошибку о неправильном расположении БД. Это было реализовано аналогично с редактором свойств TIBDatabase. Но в данном случае теперь не нужно вводить такие параметры как имя пользователя, пароль и кодировку. Нужно лишь указать расположение файла БД и имя сервера, если таковой имеется (см. рисунок 3.27).

```
void __fastcall TNastroiki::Button1Click(TObject *Sender)
{
DM->IBDatabase1->Params->Clear();
DM->IBDatabase1->LoginPrompt = false;
DM->IBDatabase1->DatabaseName = Edit1->Text + ":" + Edit2->Text;
DM->IBDatabase1->Params->Add("user_name=SYSDBA");
DM->IBDatabase1->Params->Add("password=masterkey");
DM->IBDatabase1->Params->Add("lc_ctype=win1251");
try {
DM->IBDatabase1->Connected = true;
DM->VID->Active = true;
DM->MED->Active = true;
DM->ED->Active = true;
DM->PROIZV->Active = true;
DM->SOTR->Active = true;
DM->POST->Active = true;
DM->PRIHOD->Active = true;
DM->RASHOD->Active = true;
DM->DOLJNOSTI->Active = true;
DM->MESTO->Active = true;
StatusBar1->Panels->Items[1]->Text = Edit2->Text;
ShowMessage("Успешно");
} catch (Exception &e)
{ShowMessage("Не удалось подключиться");}
}
```

Рисунок 3.27 - Программная реализация соединения с БД

Также чтобы при первом запуске программы не возникало ошибок необходимо проверять наличия соединения с БД. Если его нет, то отключаем все кнопки главного меню, кроме настроек и выхода.

Для удобства переноса программы с одного компьютера на другой сняли флажки с: «Use dynamic RLT» во вкладке «Linker» и «Build with runtime packages» во вкладке «Packages» в свойствах проекта. Это делается для того, что бы работать с программой было достаточно одного «exe»

файла, установленных клиентских библиотек Firebird и подходящего файла БД. Помимо этого задали название программы и значок, который будет изображать исполняемый файл. После чего сделали перекомпоновку программы. На этом разработка информационной подсистемы закончилась. Приложение полностью готово к тестированию на конкретном примере.

3.3 Описание контрольного примера реализации проекта

При запуске исполняемого «exe» файла готового приложения, появилось главное окно «Подсистема учета и продажи медицинских товаров» (см. рисунок 3.28).

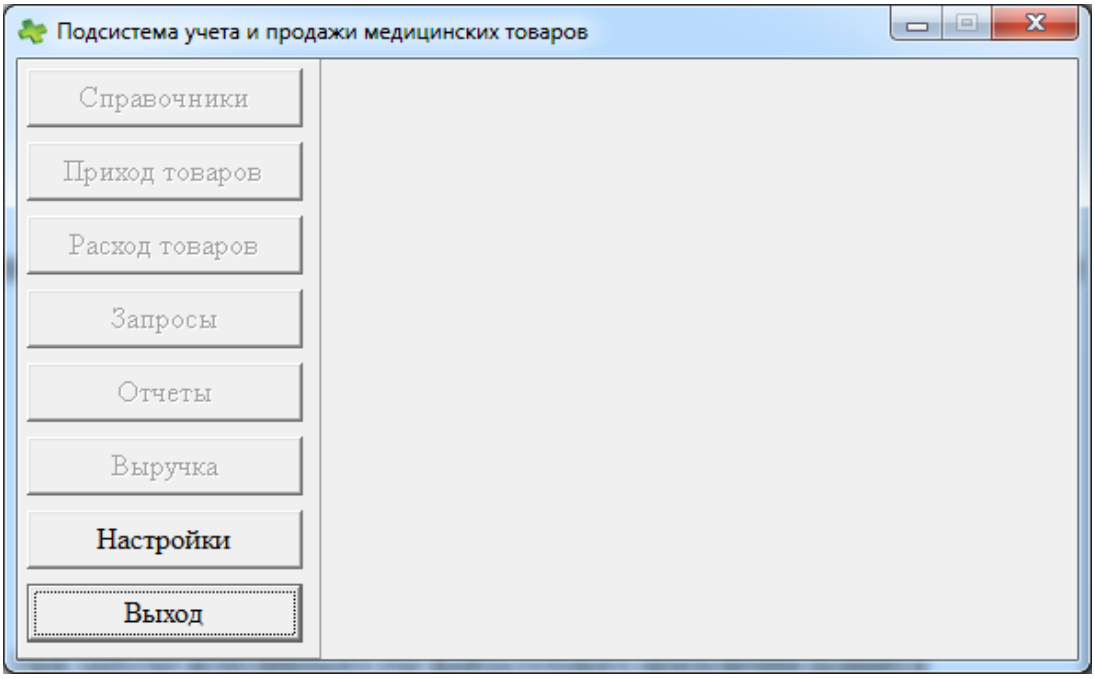

Рисунок 3.28 - Первый запуск готового приложения

Перешли в настройки приложения (см. рисунок 3.29).

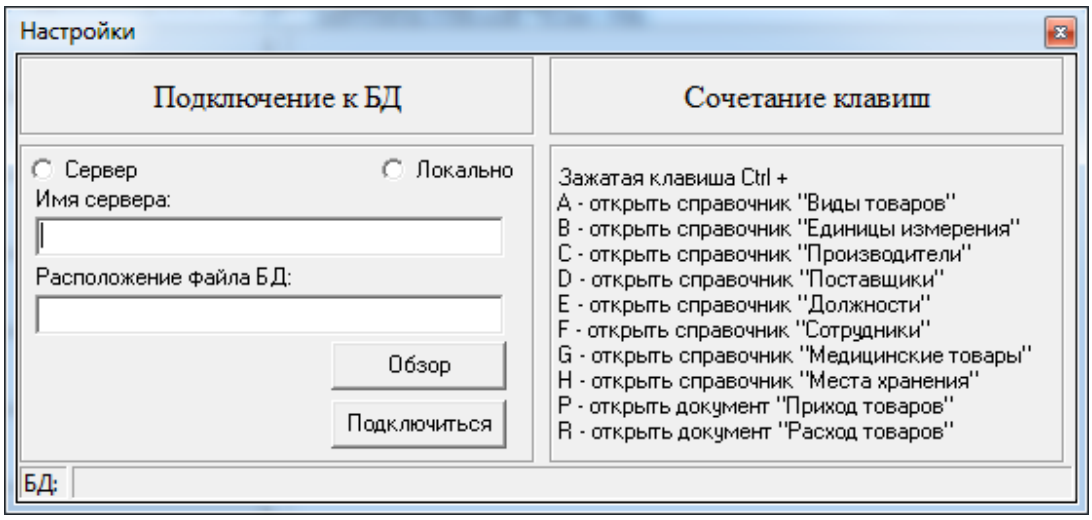

Рисунок 3.29 - Настройки программы

Создали подключение к базе данных: устанавливали переключатель на «Локально», нажимали кнопку «Обзор» и выбирали файл базы данных (см. рисунок 3.30).

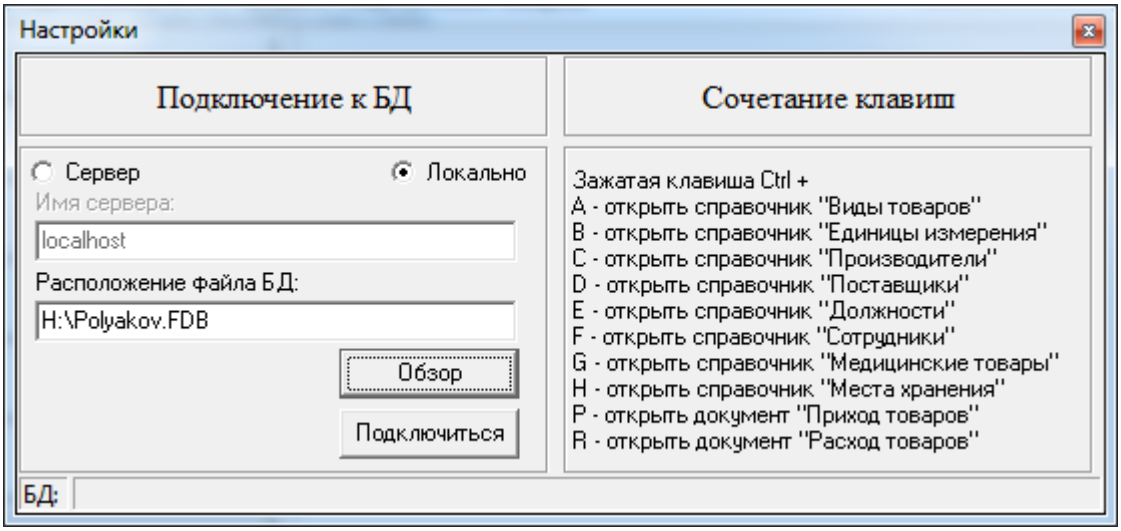

Рисунок 3.30 - Подключение к БД

Далее нажали кнопку «подключиться». Соединение прошло успешно, приложение проинформировало об этом (см. рисунок 3.31).

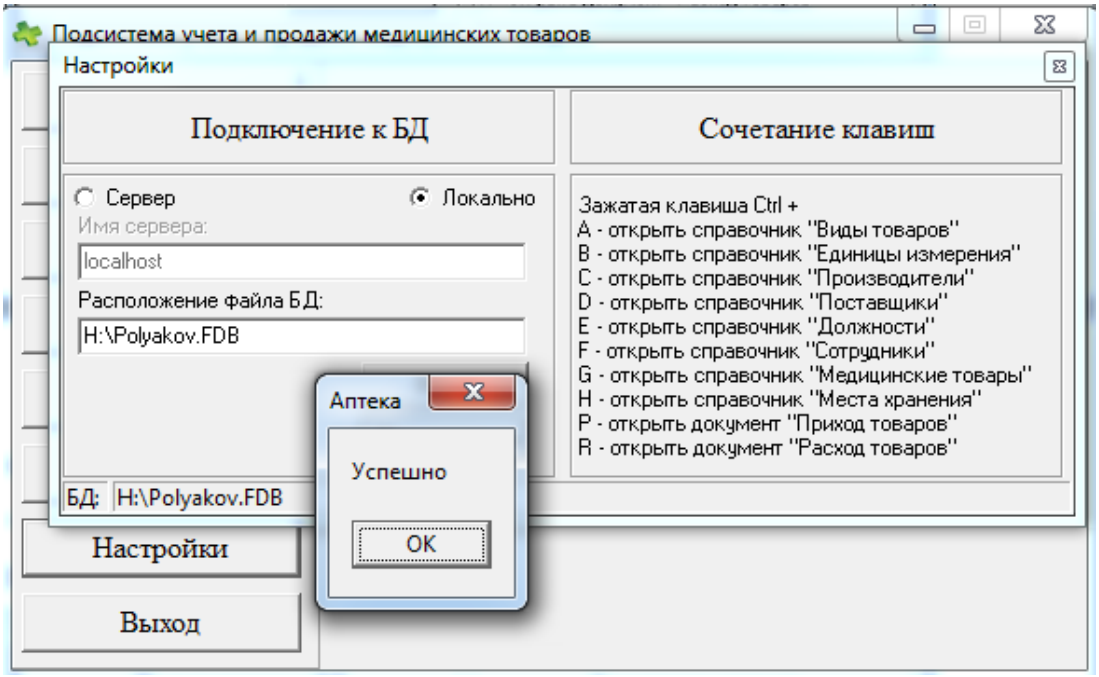

Рисунок 3.31 - Успешное соединение с БД

После входа в главное меню все кнопки стали активны. Первоначально надо заполнить все справочники. Для этого нажали кнопку справочники, выбирали нужный справочник, и переходили к его заполнению (см. рисунок 3.32). В данной главе показана не вся информация, участвующая в работе программы. Все исходные данные находятся в приложении Б.

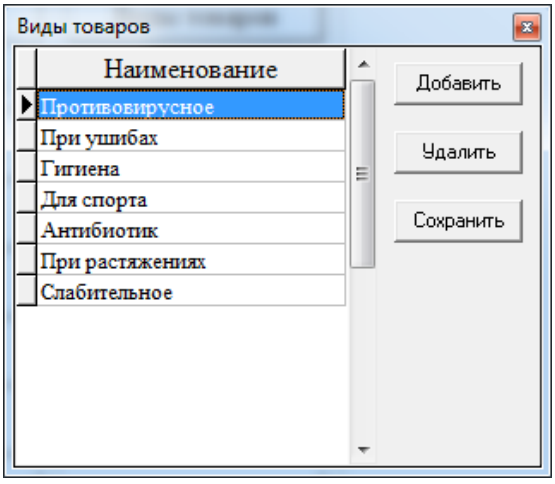

Рисунок 3.32 - Пример заполнения справочника

Аналогичным образом заполняли справочники «места хранения», «единицы измерения», «производители», «поставщики», «должности»,

«сотрудники». Перешли к заполнению справочника «Список медицинских товаров». Сам справочник имеет отличный от других вид (см. рисунок 3.33).

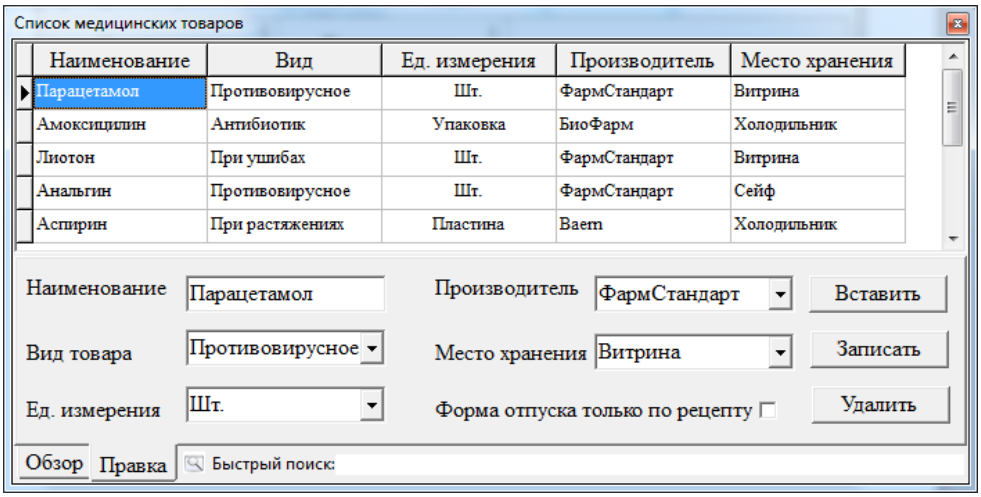

Рисунок 3.33 - Окно справочника «Список медицинских товаров»

Для его заполнения нажали кнопку «Вставить». Добавляется новая строка в таблицу, поля для ввода очищаются, курсор устанавливается в поле ввода «наименование» (см. рисунок 3.34).

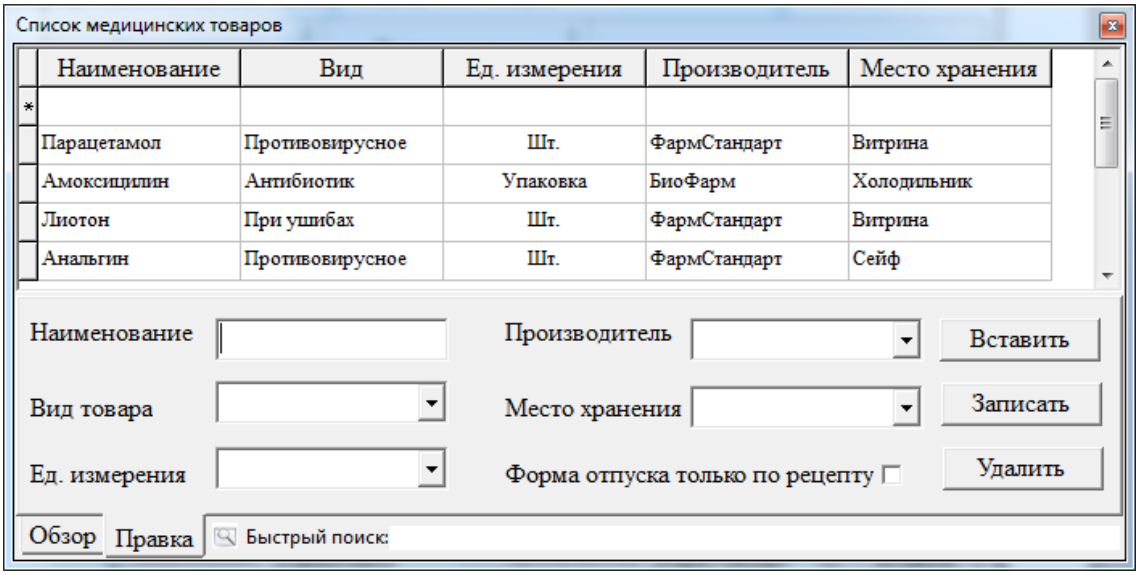

Рисунок 3.34 – Запись данных в справочник «Список медицинских товаров»

Вписав название товара, поочередно выбирали значения из выпадающих списков, а также устанавливали флажок в «Форма отпуска только по рецепту» если это необходимо. Нажали кнопку «Записать».

Программа попросит подтвердить внесение записи. Нажимали кнопку «Да» (см. рисунок 3.35).

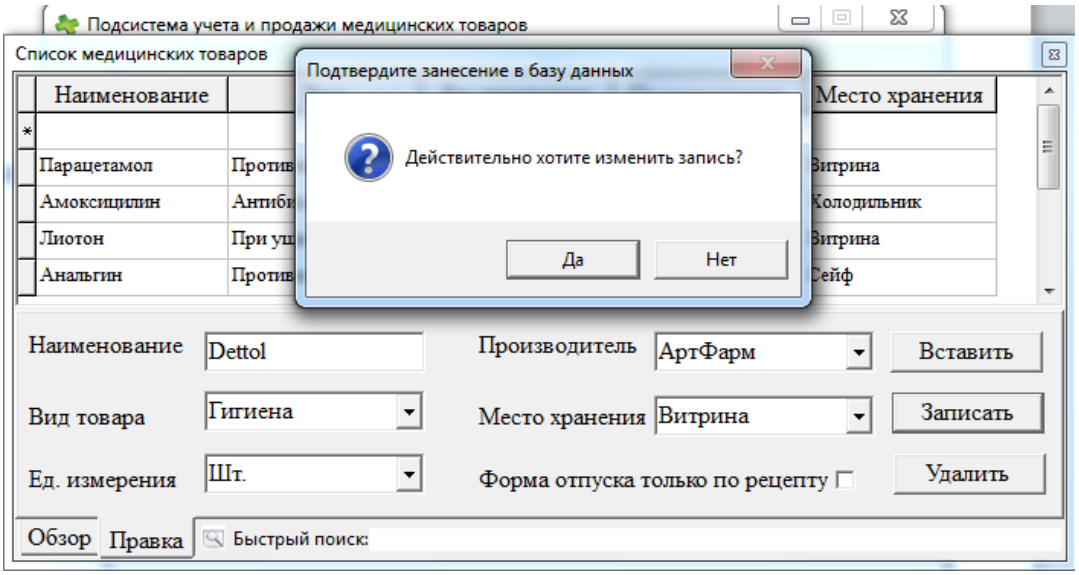

Рисунок 3.35 - Подтверждение о внесении изменений

Далее перешли во вкладку «Обзор». Нашли только что внесенный товар. Для этого воспользовались быстрым поиском, расположенным в нижней части формы. Изменили аннотацию к этому товару. Для этого нажали кнопку «Изменить» и внесли необходимую информацию о товаре», после чего нажали кнопку «Сохранить» (см. рисунок 3.36).

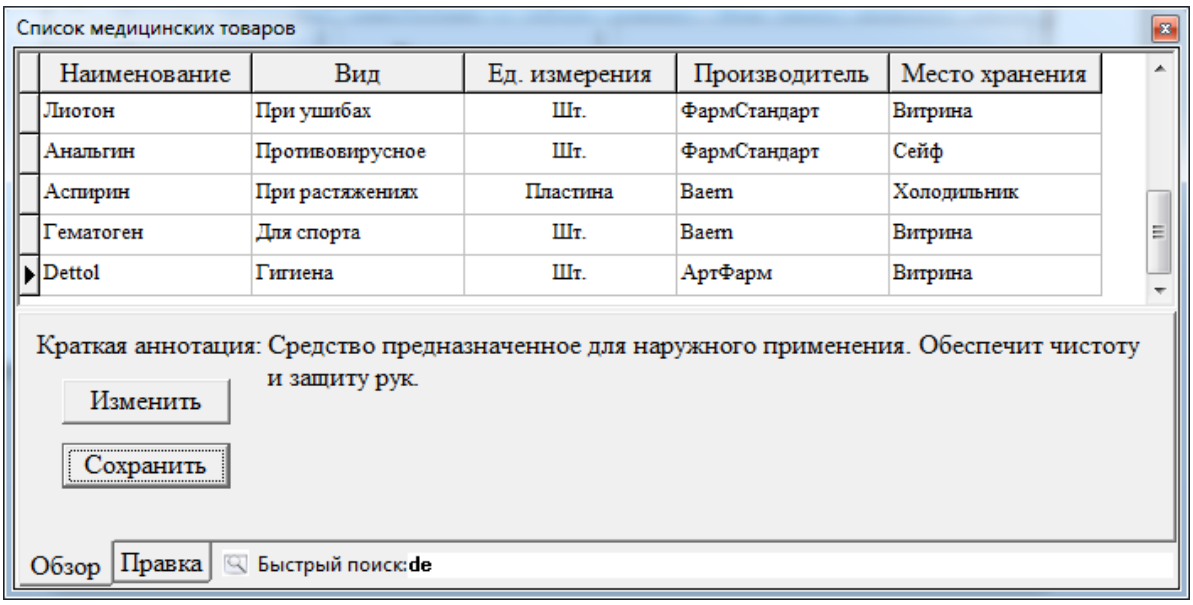

Рисунок 3.36 - Изменение аннотации о товаре

На следующем шаге оформили приход товаров. Для этого в главном окне программы нажали сочетание клавиш «Ctrl» + «P». Открылось окно «Приход товаров» (см. рисунок 3.37).

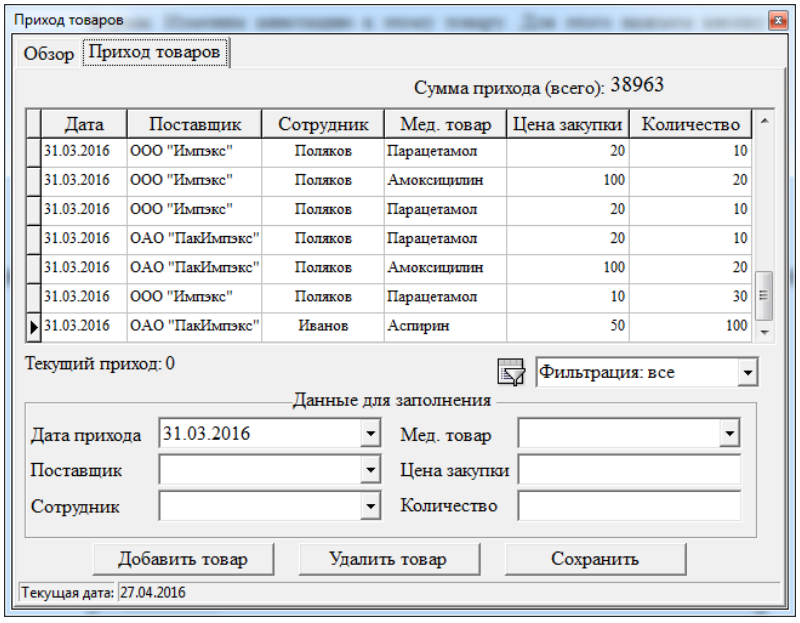

Рисунок 3.37 - Окно документа «Приход товаров»

Чтобы добавить новый товар необходимо заполнить «Данные для заполнения». Занесли информацию, при этом можно отфильтровать все товары (см. рисунок 3.38).

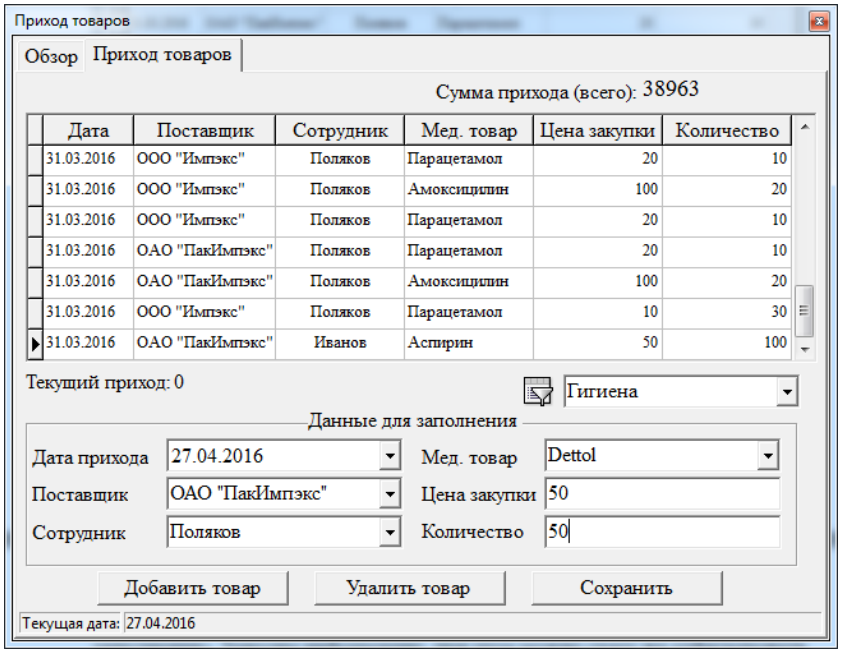

Рисунок 3.38 - Заполнение данных прихода

Нажали кнопку «Добавить товар». При выходе из формы «Приход товаров» нажимаем кнопку «Сохранить».

Далее оформили продажу товаров. Для этого в главном окне программы нажали кнопку «Расход товаров». Открылась форма «Расход товаров» (см. рисунок 3.39).

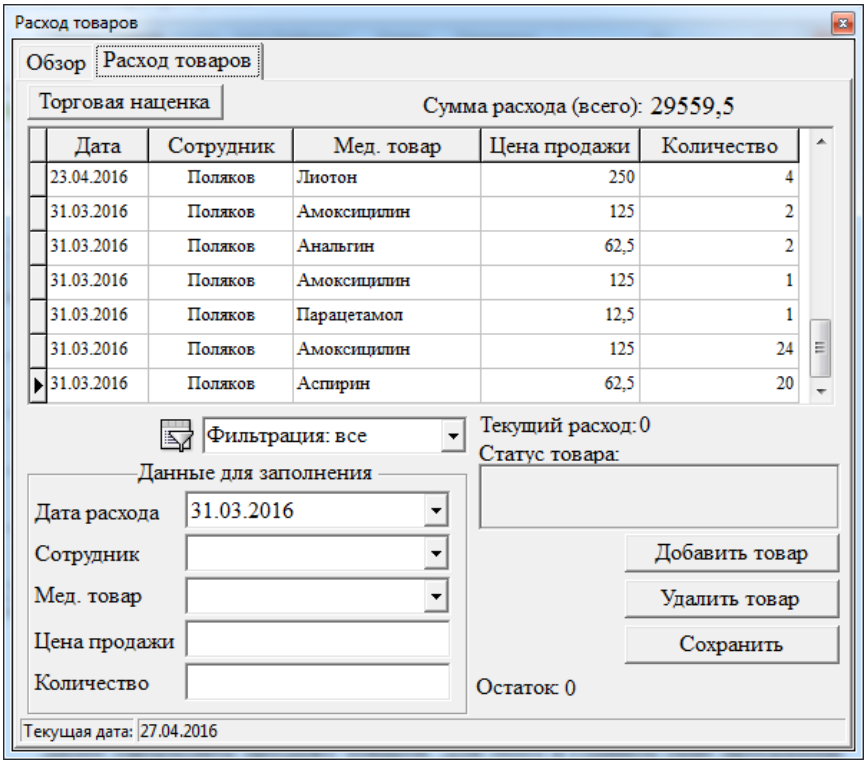

Рисунок 3.39 - Окно документа «Расход товаров»

Отфильтровать все товары можно воспользовавшись встроенной фильтрацией данных. Для продажи товара необходимо заполнить «Данные для заполнения». Внесли всю необходимую информацию. Сразу же можно проверить статус товара и его остаток (см. рисунок 3.40).

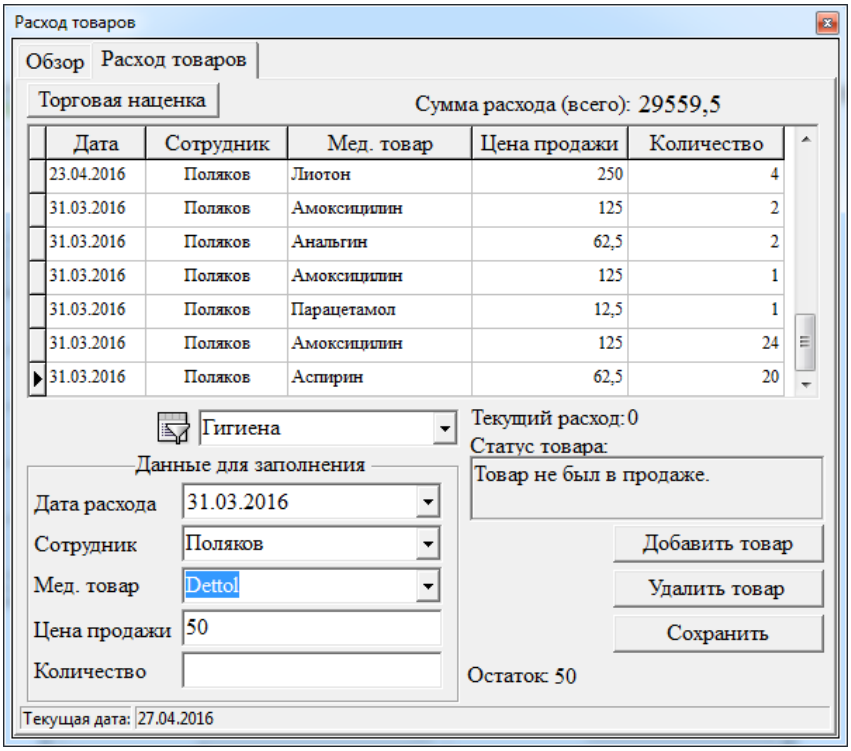

Рисунок 3.40 - Текущий остаток товара

Увидели, что товар еще не был продаже и его остаток равен 50. В верхней части окна открыли «Торговая наценка» и установили наценку равную 30%. Продали три таких товара (см. рисунок 3.41).

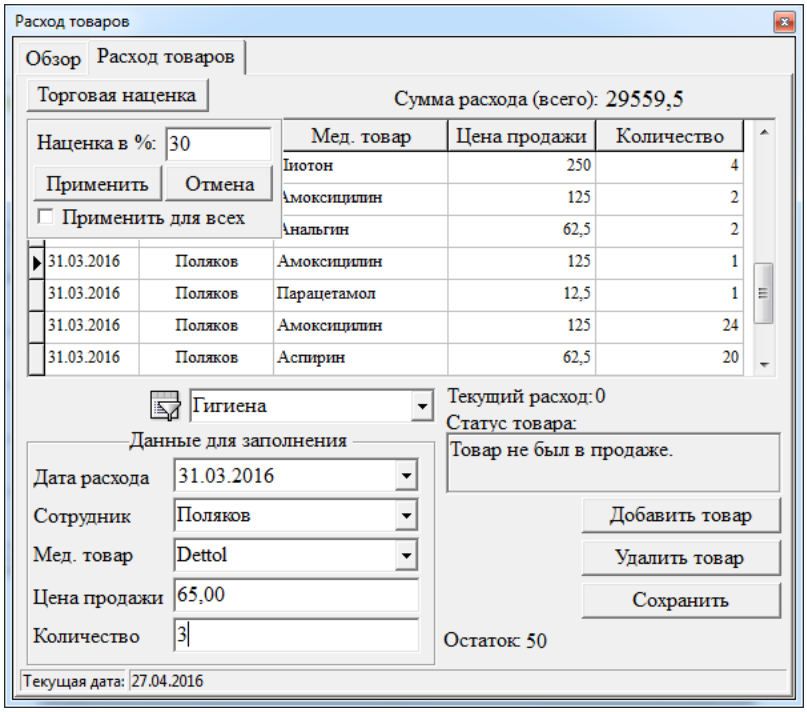

Рисунок 3.41 - Оформление расхода товаров

После этого заметили, что поменялся статус и остаток товара (см. рисунок 3.42).

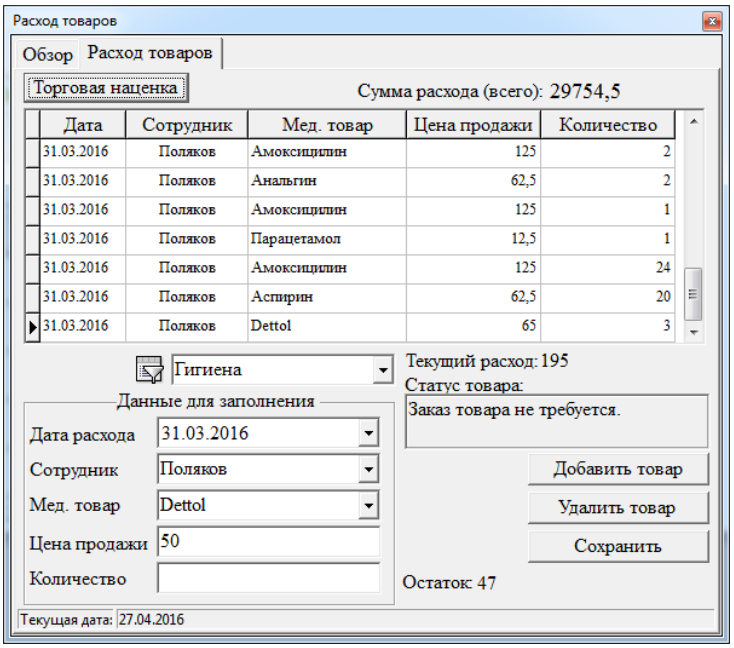

Рисунок 3.42 - Остаток товара после продажи

Перед выходом из формы «Расход товаров» нажали кнопку сохранить.

Далее выяснили, какое количество товаров было куплено за определенный период. В главном меню нажали «Запросы» и далее «Приход по дате» (см. рисунок 3.43).

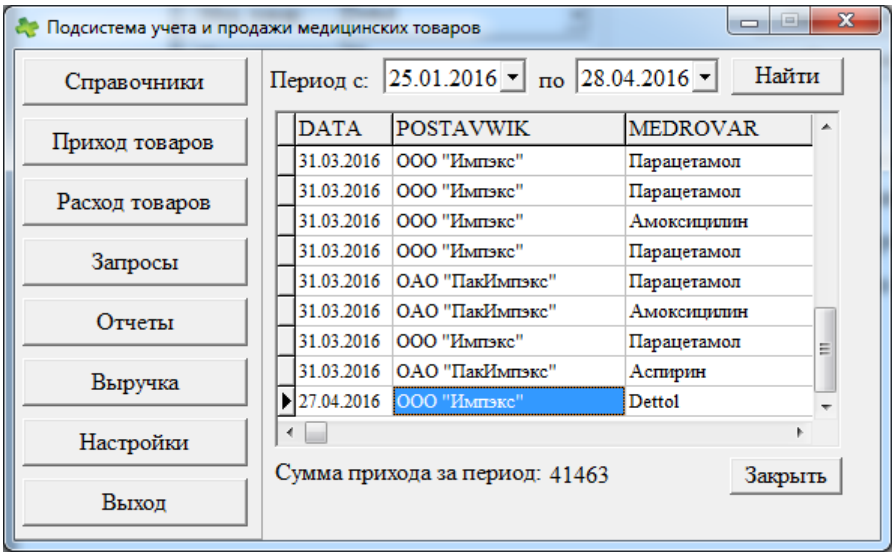

Рисунок 3.43 - Приход товаров за период

Аналогичным образом можно узнать количество товаров, которое было продано за определенный период.

И наконец, сформировали отчет обо всех товарах, имеющихся на данный момент (см. рисунок 3.44).

Список медицинских товаров
по состоянию на 27.04.2016

Наименование	Вид	Ед. измерения	Место хранения	Производитель
Парацетамол	Противовирусное	Шт.	Витрина	ФармСтандарт
Амоксицилин	Антибиотик	Упаковка	Холодильник	БиоФарм
Лиотон	При ушибах	Шт.	Витрина	ФармСтандарт
Анальгин	Противовирусное	Шт.	Сейф	ФармСтандарт
Аспирин	При растяжениях	Пластина	Холодильник	Baern
Гематоген	Для спорта	Шт.	Витрина	Baern
Dettol	Гигиена	Шт.	Витрина	АртФарм

Рисунок 3.44 - Отчет по товарам

После окончания работы нажали кнопку «Выход». Твердые копии экранов подтверждают правильную работоспособность, тем самым программный продукт полностью готов к использованию.

3.4 Оценка экономической эффективности

Разработанное программное решение необходимо рассмотреть с экономической точки зрения. Нужно выявить её экономическую целесообразность и пользу, которую она может представлять. Для этого необходимо сделать расчет некоторых экономических показателей и на их основе можно будет сделать вывод о целесообразности внедрения данного продукта.

Для оценки экономической эффективности необходимо рассчитать:

– экономию эксплуатационных расходов;

– капитальные вложения;

– срок окупаемости.

Расчет экономии эксплуатационных расходов основан на сравнении двух вариантов: базового и машинного. Для данного проекта базовым вариантом является ручной способ составления отчетности. Отчеты составляются ежемесячно. На составление всей отчетной документации ручным способом в среднем тратится 5 рабочих дней, при этом учитывается время на сбор всех необходимых приходных и расходных документов, хранящихся в бумажном виде. Как правило, формированием отчетов занимается один провизор, имеющий соответствующую квалификацию. Кроме того, при ручном способе возникают ошибки, исправление которых среднем занимает 8 часов. Трудоемкость работ по составлению отчетных документов ручным способом в год составит 576 человеко-часов.

На подготовку документов машинным способом в среднем тратится 1,5 часа, при условии своевременного и методичного ввода в систему данных из приходных и расходных документов. Получением отчетов машинным способом может быть осуществлено одним оператором, имеющим низкую квалификацию. Трудоемкость работ по составлению отчетных документов машинным способом в год составит 498 человеко-часов.

Экономический эффект от внедрения проекта определяется по формуле:

$$Э=C_1-C_2, \quad (3.1)$$

где C_1 и C_2 – затраты соответственно базового (ручного) и нового (машинного) способа решения задачи.

Затраты для базового способа рассчитываются по формуле:

$$C_1=C_з+C_с+C_а+C_{эл}, \quad (3.2)$$

где $C_з$ – заработная плата работников;

$C_с$ – затраты на социальные нужды;

$C_а$ – амортизационные отчисления на компьютер;

$C_{эл}$ – затраты на электроэнергию.

Заработная плата работников определяется по формуле:

$$C_З = \frac{O * T_P}{T_M}, \quad (3.3)$$

где О – должностной оклад специалиста за месяц;

T_P – количество рабочих часов на составление отчетной документации;

T_M – количество рабочих часов в месяц.

При 5-дневной рабочей неделе и 8-часовом рабочем дне:

$$C_З = \frac{10000 * 576}{168} = 34286 \text{ руб.}$$

Затраты на социальные нужды:

$$C_с = C_з * i, \quad (3.4)$$

где $C_з$ – затраты на оплату труда;

i – ставка единого социального налога.

$$C_с = 34286 * 0{,}26 \approx 8915 \text{ руб.}$$

Амортизационные отчисления:

$$C_а = (C_к / T_а) * T_{исп}, \quad (3.5)$$

где $T_а$ – период амортизации в часах;

$T_{исп}$ – время использования в часах;

$C_к$ – стоимость ЭВМ в руб.

Компьютер используется для составления отчетной документации: 40 часов в месяц, 576 часов в год.

$T_а$ = 5 лет, период амортизации в часах составляет 10080 часов.

$$T_{исп} = 576 \text{ часов}$$

$$C_к = 16907 \text{ руб.}$$

$$C_а = (16907 / 10080) * 576 \approx 966 \text{ руб.}$$

Затраты на электроэнергию:

$$C_{эл} = k * S * j, \quad (3.6)$$

где k – количество кВт/час, затраченное на составление отчетов ручным способом;

S – стоимость 1 кВт/час.

Персональный компьютер потребляет 0,25 кВт/час.

$$S = 3,5 \text{ руб.}$$

$$C_{эл} = 0,25 * 576 * 3,5 * 1 \approx 504 \text{ руб.}$$

Полные затраты на составление отчетов ручным способом составят:

$$C_1 = 34286 + 8915 + 966 + 504 = 44671 \text{ руб.}$$

Затраты для решения задачи машинным способом рассчитываются по формуле аналогичной для расчета по базовому способу.

При 5-дневной рабочей неделе и 8-часовом рабочем дне затраты на оплату труда составит:

$$C_з = \frac{4000 * 498}{168} = 11857 \text{ руб.}$$

Затраты на социальные нужды:

$$C_с = 11857 * 0,26 \approx 3083 \text{ руб.}$$

Амортизационные отчисления:

Компьютер используется 41,5 часов в месяц, 498 часов в год.

$T_а$ = 5 лет, период амортизации в часах составляет 10080 часов.

$$T_{исп} = 498 \text{ часов}$$

$$C_к = 16907 \text{ руб.}$$

$$C_а = (16907 / 10080) * 498 \approx 835 \text{ руб.}$$

Затраты на электроэнергию:

$$C_{эл} = 0,25 * 498 * 3,5 * 1 \approx 435 \text{ руб.}$$

Затраты при предлагаемом способе решения задачи составляют:

$$C_2 = 11857 + 3083 + 835 + 435 = 16210 \text{ руб.}$$

Экономия эксплуатационных затрат определяется путем сопоставления базового и нового способов решения задачи:

$$Э = 44671 – 16210 = 28461 \text{ руб.}$$

Капитальные вложения на создания данного продукта можно разделить:

– затраты на программное обеспечение (сумма затрат 13700 руб.);

– затраты на материалы и оборудование (сумма затрат 17000 руб.);

– затраты на оплату труда. Расчет заработной платы осуществляется на основе месячного должностного оклада разработчика программного продукта, равного 4700 руб., с учетом количества рабочих часов, затраченных на разработку (сумма затрат 8952 руб.).

Сумма первоначальных инвестиций в проект равняется 39652 рубля.

Срок окупаемости в данном случае составит 1,4 года.

На основе проделанных расчетов, можно сделать вывод, что проект эффективен с экономической точки зрения, в первую очередь благодаря низкой стоимости разработки самого проекта. Не менее важным плюсом проекта является экономия денежных средств, образующаяся благодаря значительному сокращению времени обработки и составления документации. Учитывая сумму затрат на создание проекта, а также сумму сэкономленных средств, можно рассчитать, что период окупаемости будет равен 1,4 года, а это говорит о том, что через этот небольшой промежуток времени предприятие сможет выйти на более новый уровень доходности.

ЗАКЛЮЧЕНИЕ

В выпускной квалификационной работе была рассмотрена проблема автоматизации учета и продажи медицинских товаров. Результатом работы является готовое приложение. Для решения поставленной цели были решены следующие задачи:

– изучена организационная структура предприятия;

– выявлены недостатки существующей организации обработки информации;

– обосновано принятие проектных решений;

– спроектирована и разработана база данных;

– создан программный продукт;

– протестирован созданный продукт;

– оценена эффективность работы предприятия внедряемым программным продуктом.

Разработанное программное обеспечение позволяет:

– вести базу данных медицинских товаров;

– снизить трудовые затраты;

– сократить время обработки информации;

– повысить скорость работы с документами;

– устранить многократное дублирование информации;

– максимально сократить количество бумажных документов;

– облегчить получение различных отчетов.

– повысить скорость и качество обслуживания;

– хранить информацию в удобном для использования виде;

Информационная подсистема представляет собой: локально или удалено расположенный файл базы данных и исполняемое приложение. Сама программа учитывает все аспекты ведения учета медицинских товаров и их продажи. Она может быть использована и другими организациями, занимающимися подобным видом деятельности.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Михелёв, В.М. Базы данных и СУБД [Текст] / В.М. Михелёв– Белгород: Издательство БелГУ, 2012. – 200 с.
2. Скотт Мейерс, Effective Modern C++ [Текст] / Мейерс – Москва: Бином-Пресс 2016. – 350 с.
3. Автоматизированные информационные технологии в экономике [Текст]: Учебник/ Под ред. Г.А. Титоренко – М.: Компьютер, ЮНИТИ, 2015. – 400 с.
4. Архангельский, А.Я. C++ Builder 6. Справочное пособие. [Текст] /А.Я. Архангельский - М.: Бином-Пресс, 2014. – 824 с.
5. Алан, Р. Саймон. Стратегические технологии баз данных: менеджмент на 2010 год [Текст] / Пер. с англ и предисл. М.Р. Когаловского. - М.: Финансы и статистика, 2009. – 625 с.
6. Баронов, В.В., Калянов Г.Н., Попов Ю.Н. - Автоматизация управления предприятием [Текст] / В.В. Баронов - М.: Инфра-М, 2013.– 543с.
7. Арсеньев, Б.П. Интеграция распределенных баз данных [Текст]/ Б.П. Арсеньев, С.А. Яковлев, СПб. Изд. «Лань», 2010 – 462с.
8. Архангельский, А. Я. C++Builder 6 Справочное пособие. Книга 2. Классы и компоненты. [Текст]/ А.Я. Архангельский, М.: Бином-Пресс, 2010. – 528 с.
9. Балдин, К.В. Уткин В.Б. Информационные системы в экономике. [Текст]/ К.В. Балдин, Уткин В.Б, Учебное пособие. – Дашков и К, 2011. – 395 с.
10. Борри Х. Firebird: руководство разработчика баз данных: Пер. с англ. - [Текст]/ Х. Борри, СПб.: БХВ-Петербург, 2010. - 1104 с.
11. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем. [Текст]/ А.М. Вендеров, М.: Финансы и статистика, 2010. – 176 с.

12. Вендров, А.М. Современные методы и средства проектирования информационных систем / А.М. Вендров, М.: Финансы и статистика, 2012. – 65 с.

13. Гахова, Н.Н. Инструментальные средства информационных систем: Учебно-методический комплекс [Электронный ресурс] / Н.Н. Гахова; НИУ БелГУ. - Белгород: НИУ БелГУ, 2012. - Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=5188

14. Гахов, Р.П. Методы и средства проектирования информационных систем и технологий: Учебно-методический комплекс [Электронный ресурс]/ Р.П. Гахов; Белгород, 2013. - Режим доступа: http://pegas.bsu.edu.ru/course/view.php

15. Голицына, О.Л. Программное обеспечение [Текст]/ О. Л. Голицына, И. И. Попов, Т. Л. Партыка. – М.: Форум, 2013. – 448 с.

16. Илюшечкин В.М., Основы использования и проектирования баз данных [Текст]/ В.М. Илюшечкин, М.: «Издательство Юрайт» 2010. -213с.

17. Ипатова, Э.Р. Методологии и технологии системного проектирования информационных систем [Текст]/ Э.Р. Ипатова, Ю.В. Ипатов, М.: Флинта, 2012. – 256 с.

18. Карпова, Т.С. Базы данных. Модели, разработка, реализация (2-е изд.) [Текст]/ Т.С. Карпова, М.: НОУ "Интуит", 2016. - 403с.

19. Козлов, А.С. Проектирование и исследование бизнес-процессов: Учебное пособие [Текст]/ А.С. Козлов, Москва: Флинта, 2011. - 268 с.

20. Конноли Т. Бегг К., Базы данных. Проектирование, реализация и сопровождение. Теория и практика. 3–е издание. [Текст]/Т. Конноли, К.Бегг, М.: Издательский дом "Вильямс", 2011. - 1440 с.

21. Культин, Н. Самоучитель C++Builder [Текст]/ Н. Культин, СПб.: БХВ-Перербург, 2011.-203 с.

22. Лавров, С.С. Программирование. Математические основы, средства, теория: учебное пособие. [Текст]/ С.С. Лавров, СПб.: БХВ-Петербург, 2011. - 320 с.

23. Ломакин, В.В. Программирование и программное обеспечение информационных технологий: Учебно-методический комплекс [Электронный ресурс] / В.В. Ломакин; НИУ БелГУ. - Белгород, 2014 Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=4462

24. Маклаков, С.В. BPwin, ERwin. CASE-средства разработки информационных систем. [Текст]/ С.В. Маклаков, М.: ДИАЛОГ-МИФИ, 2013. – 304 с.

25. Маторин, С.И. Теория систем и системный анализ: Учебно-методический комплекс [Электронный ресурс] / С.И. Маторин, О.А. Зимовец; НИУ БелГУ. - Белгород: НИУ БелГУ, 2012. - Режим доступа: http://pegas.bsu.edu.ru/course/view.php?id=4733

26. Мезенцев, К.Н. Автоматизированные информационные системы [Текст]/ К.Н. Мезенцев, М.: Академия, 2012. – 174 с

27. Муромцев, В.В. Проектирование информационных систем: Учебное пособие для студентов вузов заочной формы обучения по спец. 010502 "Прикладная информатика в экономике" / Муромцев В.В.; Рец.: В.А. Ломазов, С.И. Маторин; Федеральное агентство по образованию; Фак. КНИТ каф. прикладной информатики БелГУ; БелГУ. - Белгород: БелГУ, 2012. - 160 с.

28. Основы проектирования реляционных баз данных. [Электронный ресурс] Режим доступа: http://www.intuit.ru/goods_store/ebooks/8322, свободный.

29. Паттерсон, Д. Архитектура компьютера и проектирование компьютерных систем [Текст]/ Д. Паттерсон, Дж. Хеннесси, СПб.: Питер, 2012. – 784 с.

30. Пахомов, Б.И. C/C++ и Borland C++ Builder для начинающих. [Текст]/ Б.И. Пахомов, СПб.: БХВ-Петербург, 2011. – 640 с.

31. Пахомов, Б.И. С\С++ и Borland C++ Builder для студента. [Текст]/ Б.И. Пахомов, СПб.: БХВ-Петербург, 2013. – 448 с.

32. Петров, В.Н. Информационные системы. [Текст]/ В.Н. Петров, СПб.: Питер, 2012. – 688 с.

33. Послед Б.С. Borland C++ Builder 6. Разработка приложений баз данных [Текст]/ Б.С. Послед, – СПб.: ООО «ДиаСофтЮП», 2011. –320 с.

34. Репин, В. - Бизнес-процессы. Моделирование, внедрение, управление. [Текст]/ В. Репин, Москва: Флинта, 2013. - 480 с.

35. Ресурсы информационных систем. [Электронный ресурс]. Режим доступа: http://www.economica-upravlenie.ru/content/view/204/, свободный.

36. Смирнова, Г.Н. Проектирование экономических информационных систем. Учебное пособие. [Текст]/ Г.Н. Смирнова, М.: Высшая школа, 2012. – 428 с.

37. Титоренко, Г.А. Автоматизированные информационные технологии в экономике: учебное пособие. [Текст]/ Г.А. Титоренко, М.:Атлас, 2013 г. – 245 с.

38. Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий: учебное пособие. [Текст]/ Н. В. Федоров.- МГИУ, 2010.-128 с.

39. Федорова, Е.Н. Теоретические основы программирования: учебное пособие. / Е. Н. Федорова. МГИУ, 2012.-214 с.

40. Фельдман, Я.А. Создаем информационную систему [Текст]/ Я.А. Фельдман. М.: Солон-Пресс, 2011. – 120 с.

41. Хомоненко, А.Д. Базы данных: учебник для высших учебных заведений, 4-е издание дополненное и переработанное. [Текст]/ А.Д. Хомоненко, СПб.: Корона, 2012. – 736 с.

42. Хомоненко, А.Д., Ададуров С.Е. Работа с базами данных в C++ Builder. [Текст]/ А.Д. Хомоненко, СПб.:БХВ-Петербург, 2011.-496 с.

43. Беспалов, Р.С. Инструментарий разработчика бизнес-процессов. [Текст] – М.: Актион-Медиа, 2013.– 400 с.

44. Болтенков, В.И. А.Л.Литвинов, Лычёва Н.В. Конфигурирование и настройка автоматизированных информационных систем [Текст]: Учеб. пособие. - Белгород: Издательство БелГУ, 2012. – 25 с.

45. Вендров, А.М. Проектирование программного обеспечения экономических информационных систем [Текст]: Учебник. – М.: Финансы и статистика, 2012. - 352 с.

46. Годин, В.В. Корнеев И.К. - «Информационное обеспечение управленческой деятельности» [Текст]. Мск. Изд. «Высшая школа», 2012 г. – 345 с.

47. Дейт, К. Дж. Введение в системы баз данных. [Текст] - 6-е изд. - М., СПб., Киев, Изд. дом Вильяме, 2014. – 234 с.

48. Ефимова, О.Л. Технология проектирования и внедрения информационных систем - интегрированная технология ARIS [Текст] // Реинжиниринг бизнес-процессов предприятий на основе современных информационных технологий: Сб. научных трудов 3-й Российской научно-практической конференции. - М.: МЭСИ, 2015. – 124 с.

49. Жиляков, Е.Г. Методические указания по выполнению выпускной квалификационной работы по специальности 071900 «Информационные системы в экономике» [Текст] - Белгород: Издательство БУПК, 2009. – 450с.

50. Зайцева, О. А. Радугин А. А., Радугин К. А., Рогачева Н. И.. Основы менеджмента [Текст]: Учеб. пособ. – М.: Центр, 2013. – 323 с.

ПРИЛОЖЕНИЕ А

ПРОГРАММНЫЙ КОД

«DataModule.cpp»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "DataModule.h"
#include "MedTovari.h"
#include "MainForm.h"
#include "Edinici.h"
#include "VidTovara.h"
#include "Proizvoditeli.h"
#include "Postavwiki.h"
#include "Doljnosti.h"
#include "Sotrudniki.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "ViruchkaMain.h"
#include "MestoHraneniya.h"
#include "ZaprosPrihod.h"
#include "ZaprosRashod.h"
#include "Ystanovki.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TDM *DM;
//---------------------------------------------------------------------------
__fastcall TDM::TDM(TComponent* Owner)
        : TDataModule(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TDM::MEDAfterSl(TDataSet *DataSet)
{
SpMedTovari->Edit1->Text = MEDNAME->Value;
SpMedTovari->CB->ItemIndex = SpMedTovari->CB->Items-
>IndexOf(MEDVID->AsString);
SpMedTovari->CB2->ItemIndex = SpMedTovari->CB2->Items-
>IndexOf(MEDEDINICA_IZMERENIYA->AsString);
SpMedTovari->CB3->ItemIndex = SpMedTovari->CB3->Items-
>IndexOf(MEDPROIZVODITEL->AsString);
SpMedTovari->CB4->ItemIndex = SpMedTovari->CB4->Items-
>IndexOf(MEDMESTO_HRANENIYA->AsString);
if (MEDFORMA_OTPUSKA->Value == "Рецепт") SpMedTovari->CheckBox1-
>Checked = true;
else  SpMedTovari->CheckBox1->Checked = false;
}
//---------------------------------------------------------------------------
void __fastcall TDM::MEDBeforePost(TDataSet *DataSet)
{
if (!SpMedTovari->canpost) {
DataSet->Cancel();
Abort;}
}
//---------------------------------------------------------------------------
```

```
void __fastcall TDM::PRIHODAfterOpen(TDataSet *DataSet)
{
summ = 0, proizv = 1;
PRIHOD->First();
while (!PRIHOD->Eof) {
proizv = PRIHODCENA_ZAKYPKI->Value * PRIHODKOLVO->Value;
summ = summ + proizv;
PRIHOD->Next();
}
}
//---------------------------------------------------------------------------
void __fastcall TDM::RASHODAfterOpen(TDataSet *DataSet)
{
summ2 = 0, proizv2 = 1;
RASHOD->First();
while (!RASHOD->Eof) {
proizv2 = RASHODCENA_PRODAJI->Value * RASHODKOLVO->Value;
summ2 = summ2 + proizv2;
RASHOD->Next();
}
}
//---------------------------------------------------------------------------
```

«Doljnosti.cpp»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Doljnosti.h"
#include "DataModule.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Postavwiki.h"
#include "Proizvoditeli.h"
#include "VidTovara.h"
#include "MestoHraneniya.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpDoljnosti *SpDoljnosti;
//---------------------------------------------------------------------------
__fastcall TSpDoljnosti::TSpDoljnosti(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpDoljnosti::Button3Click(TObject *Sender)
{
if (DM->DOLJNOSTI->Modified) DM->DOLJNOSTI->Post();
}
//---------------------------------------------------------------------------
void __fastcall TSpDoljnosti::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
```

```
DM->DOLJNOSTI->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpDoljnosti::Button1Click(TObject *Sender)
{
DM->DOLJNOSTI->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpDoljnosti::DBGrid1TitleClick(TColumn *Column)
{
if (DM->DOLJNOSTI->IndexName == "D_SORT") DM->DOLJNOSTI-
>IndexName = "UNQ1_DOLJNOST";
else DM->DOLJNOSTI->IndexName = "D_SORT";
}
//---------------------------------------------------------------------------
```

«Edinici.cpp»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Edinici.h"
#include "DataModule.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "VidTovara.h"
#include "Doljnosti.h"
#include "MestoHraneniya.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpEdinici *SpEdinici;
//---------------------------------------------------------------------------
__fastcall TSpEdinici::TSpEdinici(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpEdinici::Button1Click(TObject *Sender)
{
DM->ED->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpEdinici::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->ED->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpEdinici::Button3Click(TObject *Sender)
{
```

```
if (DM->ED->Modified) DM->ED->Post();
}
//---------------------------------------------------------------------------
void __fastcall TSpEdinici::DBGrid1TitleClick(TColumn *Column)
{
if (DM->ED->IndexName == "ED_SORT") DM->ED->IndexName =
"UNQ1_EDINICI_IZMERENIYA";
else DM->ED->IndexName = "ED_SORT";
}
//---------------------------------------------------------------------------
                                        «MainForm.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "MainForm.h"
#include "DataModule.h"
#include "Edinici.h"
#include "MedTovari.h"
#include "VidTovara.h"
#include "Proizvoditeli.h"
#include "Postavwiki.h"
#include "Doljnosti.h"
#include "Sotrudniki.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "ViruchkaMain.h"
#include "MestoHraneniya.h"
#include "Ystanovki.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma link "ZaprosPrihod"
#pragma link "ZaprosRashod"
#pragma resource "*.dfm"
TAptekaMain *AptekaMain;
//---------------------------------------------------------------------------
__fastcall TAptekaMain::TAptekaMain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::FormCreate(TObject *Sender)
{
s = 0;
panel[0] = Spravochniki;
panel[1] = Zaprosi;
panel[2] = Reports;
Spravochniki->SendToBack();
Zaprosi->SendToBack();
Reports->SendToBack();
max_left = 165;
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button8Click(TObject *Sender)
{ SpVidTovara->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button9Click(TObject *Sender)
{
SpEdinici->Show();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button10Click(TObject *Sender)
{ SpProizvoditeli->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button11Click(TObject *Sender)
{ SpPostavwiki->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button12Click(TObject *Sender)
{ SpDoljnosti->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button14Click(TObject *Sender)
{ SpMedTovari->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button13Click(TObject *Sender)
{ SpSotrudniki->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button2Click(TObject *Sender)
{ PrihodMain->Show();}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button3Click(TObject *Sender)
{ RashodMain->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button7Click(TObject *Sender)
{ Close(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button17Click(TObject *Sender)
{ RepMedT->RepMed->Preview(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button6Click(TObject *Sender)
{ Viruchka->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button16Click(TObject *Sender)
{ SpMesto->Show(); }
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Timer1Timer(TObject *Sender)
{
int i;
if (panel[s]->Tag == 2) panel[s]->Tag = 0;

for (i = 0; i < 3; i++) {

if (panel[i]->Tag == 1) {
 if (panel[i]->Left != max_left)
  panel[i]->Left = panel[i]->Left + 15;
 else {Timer1->Enabled = false; panel[i]->Tag = 2; s = i;}
 }
else { if (panel[i]->Tag == 0)  {
        if (panel[i]->Left != 0) panel[i]->Left = panel[i]->Left - 15;
        else { panel[i]->Tag = 0;}
      }
      }
      }
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button1Click(TObject *Sender)
{
if (Spravochniki->Tag == 2) Spravochniki->Tag = 0;
else Spravochniki->Tag = 1 ;
Timer1->Enabled = true;
}
```

```
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button4Click(TObject *Sender)
{
if (Zaprosi->Tag == 2) Zaprosi->Tag = 0;
else Zaprosi->Tag = 1 ;
Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button5Click(TObject *Sender)
{
if (Reports->Tag == 2) Reports->Tag = 0;
else Reports->Tag = 1 ;
Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::ZpPrihod1Button2Click(TObject *Sender)
{
ZpPrihod1->Visible = false;
ZpPrihod1->Label4->Caption = "0";
DM->ZaprosP->Close();
DM->ZaprosP->SQL->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button15Click(TObject *Sender)
{
ZpRashod1->Visible = false;
ZpPrihod1->Visible = true;
ZpPrihod1->SendToBack();
Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::ZpRashod1Button2Click(TObject *Sender)
{
ZpRashod1->Visible = false;
ZpRashod1->Label4->Caption = "0";
DM->ZaprosR->Close();
DM->ZaprosR->SQL->Clear();
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button18Click(TObject *Sender)
{
ZpPrihod1->Visible = false;
ZpRashod1->Visible = true;
ZpRashod1->SendToBack();
Timer1->Enabled = true;
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::FormKeyDown(TObject *Sender, WORD &Key,
      TShiftState Shift)
{
if ((Key == 'A')&& Shift.Contains(ssCtrl)){
Button8Click(Button8); return;}
if ((Key == 'B')&& Shift.Contains(ssCtrl)){
Button9Click(Button9); return;}
if ((Key == 'C')&& Shift.Contains(ssCtrl)){
Button10Click(Button10); return;}
if ((Key == 'D')&& Shift.Contains(ssCtrl)){
Button11Click(Button11); return;}
if ((Key == 'E')&& Shift.Contains(ssCtrl)){
Button12Click(Button12); return;}
```

```
if ((Key == 'F')&& Shift.Contains(ssCtrl)){
Button13Click(Button13); return;}
if ((Key == 'G')&& Shift.Contains(ssCtrl)){
Button14Click(Button14); return;}
if ((Key == 'H')&& Shift.Contains(ssCtrl)){
Button16Click(Button16); return;}
if ((Key == 'P')&& Shift.Contains(ssCtrl)){
Button2Click(Button2); return;}
if ((Key == 'R')&& Shift.Contains(ssCtrl)){
Button3Click(Button3); return;}
}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::Button19Click(TObject *Sender)
{Nastroiki->Show();}
//---------------------------------------------------------------------------
void __fastcall TAptekaMain::FormActivate(TObject *Sender)
{
if (DM->IBDatabase1->Connected == false)
{
Button1->Enabled = false;
Button2->Enabled = false;
Button3->Enabled = false;
Button4->Enabled = false;
Button5->Enabled = false;
Button6->Enabled = false;
AptekaMain->KeyPreview = false;}
else {
Button1->Enabled = true;
Button2->Enabled = true;
Button3->Enabled = true;
Button4->Enabled = true;
Button5->Enabled = true;
Button6->Enabled = true;
AptekaMain->KeyPreview = true;;
}
}
//---------------------------------------------------------------------------
                                        «MedTovari.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "MedTovari.h"
#include "DataModule.h"
#include "MainForm.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MestoHraneniya.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "VidTovara.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpMedTovari *SpMedTovari;
//---------------------------------------------------------------------------
```

```
__fastcall TSpMedTovari::TSpMedTovari(TComponent* Owner)
      : TForm(Owner)
{
Application->OnMessage = OnApplicationMessage;
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::FormShow(TObject *Sender)
{
canpost = false;
Edit1->Clear();
CB->Clear();
DM->VID->Active = true;
DM->VID->First();
while (!DM->VID->Eof) {
CB->Items->Add(DM->VIDNAME->AsString);
DM->VID->Next(); }
CB2->Clear();
DM->ED->Active = true;
DM->ED->First();
while (!DM->ED->Eof) {
CB2->Items->Add(DM->EDNAME->AsString);
DM->ED->Next(); }
CB3->Clear();
DM->PROIZV->Active = true;
DM->PROIZV->First();
while (!DM->PROIZV->Eof) {
CB3->Items->Add(DM->PROIZVNAME->AsString);
DM->PROIZV->Next(); }
CB4->Clear();
DM->MESTO->Active = true;
DM->MESTO->First();
while (!DM->MESTO->Eof) {
CB4->Items->Add(DM->MESTONAME->AsString);
DM->MESTO->Next();}
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->MED->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Button1Click(TObject *Sender)
{
DM->MED->Insert();
Edit1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Button3Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите изменить
запись?","Подтвердите занесение в базу данных",
 MB_YESNO + MB_ICONQUESTION) == IDYES)
   {
   DM->MED->Edit();
   DM->MEDNAME->AsString = Edit1->Text;
   DM->MEDVID->AsString = CB->Text;
   DM->MEDEDINICA_IZMERENIYA->AsString = CB2->Text;
```

```
   DM->MEDPROIZVODITEL->AsString = CB3->Text;
   DM->MEDMESTO_HRANENIYA->AsString = CB4->Text;
   if (CheckBox1->Checked == true) DM->MEDFORMA_OTPUSKA->AsString
= "Рецепт";
   else DM->MEDFORMA_OTPUSKA->AsString = "Без рецепта";
   canpost = true;
   DM->MED->Post();
   }
else {
 DM->MED->Cancel();
 canpost = false; }
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::CBKeyPress(TObject *Sender, char &Key)
{Key = 0;}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::PageControl1Change(TObject *Sender)
{
if (PageControl1->ActivePage == TabSheet1)
 DM->MED->AfterScroll;
 else
 {
  DM->MED->Cancel();
  DM->MED->Refresh();
 }
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::DBGrid1KeyPress(TObject *Sender, char &Key)
{Key = 0;}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::DBGrid1TitleClick(TColumn *Column)
{
if (DM->MED->IndexName == "SORT_MED") DM->MED->IndexName =
"UNQ1_MED_TOVAR";
else DM->MED->IndexName = "SORT_MED";
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Button4Click(TObject *Sender)
{
DBRichEdit1->ReadOnly = false;
DBRichEdit1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Button5Click(TObject *Sender)
{
if (DBRichEdit1->Modified){ DM->MED->Edit();
DM->MEDANNOTACIYA->AsString = DBRichEdit1->Text;
canpost = true;
DM->MED->Post();
}
else {DM->MED->Cancel(); canpost = false;}
DBRichEdit1->ReadOnly = true;
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::CBDblClick(TObject *Sender)
{SpVidTovara->Show();}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::CB2DblClick(TObject *Sender)
{SpEdinici->Show();}
//---------------------------------------------------------------------------
```

```
void __fastcall TSpMedTovari::CB3DblClick(TObject *Sender)
{SpProizvoditeli->Show();}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::CB4DblClick(TObject *Sender)
{SpMesto->Show();}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::stbDrawPanel(TStatusBar *StatusBar,
     TStatusPanel *Panel, const TRect &Rect)
{
   TCanvas * can = StatusBar->Canvas;
   int X(Rect.Left + 2);
   int Y((stb->Height - img->Height) / 2 + 1);
   Graphics::TBitmap * bmp = new Graphics::TBitmap;
   bmp->Assign(img->Picture);
   can->Draw(X, Y, bmp);
   delete bmp;
   can->TextOutA(X + img->Width + 4, Y + 2, "");
}
//---------------------------------------------------------------------------
void __fastcall TSpMedTovari::Edit2Change(TObject *Sender)
{
TLocateOptions SearchOptions;
Variant locvalues[] = {Edit2->Text};
DM->MED->Locate("NAME", VarArrayOf(locvalues,1),
SearchOptions<<loPartialKey<<loCaseInsensitive);
}
//---------------------------------------------------------------------------
                                        «MestaHranenita.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "MestoHraneniya.h"
#include "DataModule.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "VidTovara.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpMesto *SpMesto;
//---------------------------------------------------------------------------
__fastcall TSpMesto::TSpMesto(TComponent* Owner)
      : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpMesto::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
```

```
DM->MESTO->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpMesto::Button1Click(TObject *Sender)
{
DM->MESTO->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpMesto::Button3Click(TObject *Sender)
{
if (DM->MESTO->Modified) DM->MESTO->Post();
}
//---------------------------------------------------------------------------
                                        «Postavwiki.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Postavwiki.h"
#include "DataModule.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Proizvoditeli.h"
#include "VidTovara.h"
#include "Doljnosti.h"
#include "MestoHraneniya.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpPostavwiki *SpPostavwiki;
//---------------------------------------------------------------------------
__fastcall TSpPostavwiki::TSpPostavwiki(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpPostavwiki::Button1Click(TObject *Sender)
{
DM->POST->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpPostavwiki::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->POST->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpPostavwiki::Button3Click(TObject *Sender)
{
if (DM->POST->Modified) DM->POST->Post();
}
```

```
//---------------------------------------------------------------------------
void __fastcall TSpPostavwiki::DBGrid1TitleClick(TColumn *Column)
{
if (DM->POST->IndexName == "POST_SORT") DM->POST->IndexName =
"UNQ1_POSTAVWIK";
else DM->POST->IndexName = "POST_SORT";
}
//---------------------------------------------------------------------------
```

«Prihod.cpp»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Prihod.h"
#include "DataModule.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Postavwiki.h"
#include "Proizvoditeli.h"
#include "Sotrudniki.h"
#include "VidTovara.h"
#include "MestoHraneniya.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TPrihodMain *PrihodMain;
//---------------------------------------------------------------------------
__fastcall TPrihodMain::TPrihodMain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::CBKeyPress(TObject *Sender, char &Key)
{Key = 0;}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::FormShow(TObject *Sender)
{
tsumm = 0, tproizv = 1;
TDateTime NowDate = Date();
StatusBar1->Panels->Items[1]->Text=NowDate;
Filter->Clear();
Filter->Items->Add("Фильтрация: все");
Filter->ItemIndex = 0;
DM->VID->Active = true;
DM->VID->First();
while (!DM->VID->Eof) {
Filter->Items->Add(DM->VIDNAME->AsString);
DM->VID->Next();}
CB->Clear();
CB2->Clear();
DM->POST->Active = true;
DM->POST->First();
while (!DM->POST->Eof) {
CB->Items->Add(DM->POSTNAME->AsString);
DM->POST->Next();}
DM->SOTR->Active = true;
```

```
DM->SOTR->First();
while (!DM->SOTR->Eof) {
CB2->Items->Add(DM->SOTRFAMILIYA->AsString);
DM->SOTR->Next();}
Label8->Caption = DM->summ;
FilterChange(Filter);
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Edit2KeyPress(TObject *Sender, char &Key)
{
 if ( ( Key >= '0') && ( Key <= '9' ) ) return;
 if ((Key == ',') || (Key == '.'))
    {  Key = DecimalSeparator;
       if ( (Edit2->Text).Pos(DecimalSeparator) != 0 )
          Key = 0;
       return;
    }
 if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
    {
       Edit3->SetFocus();
       return;
    }
    Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Edit3KeyPress(TObject *Sender, char &Key)
{
 if ( ( Key >= '0') && ( Key <= '9' ) ) return;
  if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
    {
       Button1->SetFocus();
       return;
    }
    Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Button1Click(TObject *Sender)
{
if (CB->Text == "" || CB2->Text == "" || CB3->Text == ""|| Edit2->Text == "" ||
Edit3->Text == "" || StrToFloat(Edit2->Text) == 0 || StrToInt(Edit3->Text) == 0)
ShowMessage("Неправильный ввод данных. Повторите попытку");
else {
   tproizv = StrToFloat(Edit2->Text) * StrToInt(Edit3->Text);
   tsumm = tsumm + tproizv;
   Label10->Caption = FloatToStr(tsumm);
   DM->PRIHOD->Insert();
   DM->PRIHOD->Edit();
   DM->PRIHODDATA->AsString = DTP->Date;
   DM->PRIHODPOSTAVWIK->AsString = CB->Text;
   DM->PRIHODSOTRUDNIK->AsString = CB2->Text;
   DM->PRIHODMEDROVAR->AsString = CB3->Text;
   DM->PRIHODCENA_ZAKYPKI->AsString = Edit2->Text;
   DM->PRIHODKOLVO->AsString = Edit3->Text;
   DM->PRIHOD->Post();
   CB3->ItemIndex = -1;
   Edit3->Clear();
```

```
   Edit2->Clear();
   DM->PRIHODAfterOpen(DM->PRIHOD);
   Label8->Caption = DM->summ;
   }
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->PRIHOD->Delete();
DM->PRIHODAfterOpen(DM->PRIHOD);
Label8->Caption = DM->summ;
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Button3Click(TObject *Sender)
{if (DM->PRIHOD->Modified) DM->PRIHOD->Post();}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::DBGrid1KeyPress(TObject *Sender, char &Key)
{Key = 0;}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::FilterChange(TObject *Sender)
{
DM->FilterTable->Close();
DM->FilterTable->SQL->Clear();
if (Filter->ItemIndex == 0)
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR");
else
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR where VID
LIKE '%"+Filter->Text+"'" );
DM->FilterTable->Open();
CB3->Clear();
DM->FilterTable->First();
while (!DM->FilterTable->Eof) {
CB3->Items->Add(DM->FilterTable->FieldByName("NAME")->AsString);
DM->FilterTable->Next();}
}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::Image1Click(TObject *Sender)
{Filter->ItemIndex = 0; FilterChange(Filter);}
//---------------------------------------------------------------------------
void __fastcall TPrihodMain::FormClose(TObject *Sender,
     TCloseAction &Action)
{
tsumm = 0; tproizv = 1;
Label10->Caption = "0";
}
//---------------------------------------------------------------------------
```

«Proizvoditeli.cpp»

```
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Proizvoditeli.h"
#include "DataModule.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "VidTovara.h"
#include "Doljnosti.h"
```

```
#include "MestoHraneniya.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpProizvoditeli *SpProizvoditeli;
//---------------------------------------------------------------------------
__fastcall TSpProizvoditeli::TSpProizvoditeli(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpProizvoditeli::Button3Click(TObject *Sender)
{
if (DM->PROIZV->Modified) DM->PROIZV->Post();
}
//---------------------------------------------------------------------------
void __fastcall TSpProizvoditeli::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->PROIZV->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpProizvoditeli::Button1Click(TObject *Sender)
{
DM->PROIZV->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpProizvoditeli::DBGrid1TitleClick(TColumn *Column)
{
if (DM->PROIZV->IndexName == "P_SORT") DM->PROIZV->IndexName =
"UNQ1_PROIZVODITEL";
else DM->PROIZV->IndexName = "P_SORT";
}
//---------------------------------------------------------------------------
                                        «Rashod.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Rashod.h"
#include "DataModule.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Sotrudniki.h"
#include "VidTovara.h"
#include "MestoHraneniya.h"
#include "ReportMedTovari.h"
```

```
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TRashodMain *RashodMain;
//---------------------------------------------------------------------------
__fastcall TRashodMain::TRashodMain(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::FormShow(TObject *Sender)
{
tsumm = 0, tproizv = 1;
TDateTime NowDate = Date();
StatusBar1->Panels->Items[1]->Text=NowDate;
Filter->Clear();
Filter->Items->Add("Фильтрация: все");
Filter->ItemIndex = 0;
DM->VID->Active = true;
DM->VID->First();
while (!DM->VID->Eof) {
Filter->Items->Add(DM->VIDNAME->AsString);
DM->VID->Next();}
CB->Clear();
CB2->Clear();
DM->SOTR->Active = true;
DM->SOTR->First();
while (!DM->SOTR->Eof) {
CB->Items->Add(DM->SOTRFAMILIYA->AsString);
DM->SOTR->Next();}
Label7->Caption = DM->summ2;
FilterChange(Filter);
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Edit1KeyPress(TObject *Sender, char &Key)
{
 if ( ( Key >= '0') && ( Key <= '9' ) ) return;
 if ((Key == ',') || (Key == '.'))
    {  Key = DecimalSeparator;
       if ( (Edit1->Text).Pos(DecimalSeparator) != 0 )
          Key = 0;
       return;
    }
 if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
    {
       Edit2->SetFocus();
       return;
    }
    Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Edit2KeyPress(TObject *Sender, char &Key)
{
if ( ( Key >= '0') && ( Key <= '9' ) ) return;
  if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
```

```
    {
      Button1->SetFocus();
      return;
    }
    Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::DBGrid1KeyPress(TObject *Sender, char &Key)
{
Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Button1Click(TObject *Sender)
{
if (CB->Text == "" || CB2->Text == "" || Edit1->Text == "" || Edit2->Text == "" ||
StrToFloat(Edit1->Text) == 0 || StrToInt(Edit2->Text) == 0)
ShowMessage("Неправильный ввод данных. Повторите попытку");
else {
 if (StrToInt(Edit2->Text) > StrToInt(Label9->Caption))
ShowMessage("Введенное количество превышает остаток");
 else {
   tproizv = StrToFloat(Edit1->Text) * StrToInt(Edit2->Text);
   tsumm = tsumm + tproizv;
   Label11->Caption = FloatToStr(tsumm);
   DM->RASHOD->Insert();
   DM->RASHOD->Edit();
   DM->RASHODDATA->AsString = DTP->Date;
   DM->RASHODSOTRUDNIK->AsString = CB->Text;
   DM->RASHODMEDTOVAR->AsString = CB2->Text;
   DM->RASHODCENA_PRODAJI->AsString = Edit1->Text;
   DM->RASHODKOLVO->AsString = Edit2->Text;
   DM->RASHOD->Post();
   CB2->ItemIndex = -1;
   Edit1->Clear();
   Edit2->Clear();
   DM->RASHODAfterOpen(DM->RASHOD);
   Label7->Caption = DM->summ2;
   Label9->Caption = "";
   }
   }

}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->RASHOD->Delete();
DM->RASHODAfterOpen(DM->RASHOD);
Label7->Caption = DM->summ2;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Button3Click(TObject *Sender)
{if (DM->RASHOD->Modified) DM->RASHOD->Post();}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::CB2Click(TObject *Sender)
{
Label9->Caption = "";
DM->Ostatki->Close();
```

```
DM->Ostatki->SQL->Clear();
DM->Ostatki->SQL->Add("select sum(kolvo) as ostatok from ostatki where
medrovar = '" + CB2->Text + "' group by ostatki.medrovar");
DM->Ostatki->Open();
if (!DM->Ostatki->FieldByName("ostatok")->IsNull)
Label9->Caption = DM->Ostatki->FieldValues["ostatok"];
else Label9->Caption = "0";
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::CB2Change(TObject *Sender)
{
Edit1->Clear();
DM->CENAP->Close();
DM->CENAP->SQL->Clear();
DM->CENAP->SQL->Add("select CENA_ZAKYPKI from PRIHOD where
medrovar = '" + CB2->Text + "'");
DM->CENAP->Open();
DM->CENAP->Last();
if (!DM->CENAP->FieldByName("CENA_ZAKYPKI")->IsNull)
Edit1->Text = DM->CENAP->FieldValues["CENA_ZAKYPKI"];
else Edit1->Text = "0";
if (CB1->Checked) BitBtn1Click(BitBtn1);

float ave;
DM->Status->Close();
DM->Status->SQL->Clear();
DM->Status->SQL->Add("select sr from status where medtovar = '" + CB2->Text
+ "'");
DM->Status->Open();
if (DM->Status->FieldByName("SR")->IsNull) Label14->Caption = "Товар не
был в продаже.";
else { ave = StrToInt(Label9->Caption) / DM->Status->FieldValues["SR"];
 if (ave < 7) Label14->Caption = "Товар закончится менее чем через 7 дней.
Необходимо сделать заказ.";
 else Label14->Caption = "Заказ товара не требуется.";
 }
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::BitBtn1Click(TObject *Sender)
{
float a,b,c;
a = StrToFloat(Edit1->Text);
b = StrToFloat(Edit3->Text);
c = a * (1 + b/100);
Edit1->Text = FloatToStrF(c,ffFixed,15,2);
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::FilterChange(TObject *Sender)
{
DM->FilterTable->Close();
DM->FilterTable->SQL->Clear();
if (Filter->ItemIndex == 0)
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR");
else
DM->FilterTable->SQL->Add("Select NAME from MED_TOVAR where VID
LIKE '%"+Filter->Text+"'" );
DM->FilterTable->Open();
CB2->Clear();
DM->FilterTable->First();
while (!DM->FilterTable->Eof) {
```

```
CB2->Items->Add(DM->FilterTable->FieldByName("NAME")->AsString);
DM->FilterTable->Next();}
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Button4Click(TObject *Sender)
{
if (GroupBox1->Visible == false) GroupBox1->Visible = true;
else GroupBox1->Visible = false;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Label4MouseMove(TObject *Sender,
      TShiftState Shift, int X, int Y)
{
Label4->Font->Color = clBlue;
Label4->Font->Style = Label4->Font->Style << fsUnderline;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Label4MouseLeave(TObject *Sender)
{
Label4->Font->Color = clBlack;
Label4->Font->Style = Label4->Font->Style >> fsUnderline;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Label4Click(TObject *Sender)
{
ShowMessage("Цена на товар установливается как: цена последний покупки *
наценку. Но ее также можно менять вручную");
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Edit3KeyPress(TObject *Sender, char &Key)
{
if ( ( Key >= '0') && ( Key <= '9' ) ) return;
 if ((Key == ',') || (Key == '.'))
    {  Key = DecimalSeparator;
       if ( (Edit3->Text).Pos(DecimalSeparator) != 0 )
          Key = 0;
       return;
    }
 if (Key == VK_BACK)
        return;
 if ( Key == VK_RETURN)
    {
       BitBtn1->SetFocus();
       return;
    }
    Key = 0;
}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::FilterKeyPress(TObject *Sender, char &Key)
{Key = 0;}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::Image1Click(TObject *Sender)
{Filter->ItemIndex = 0; FilterChange(Filter);}
//---------------------------------------------------------------------------
void __fastcall TRashodMain::FormClose(TObject *Sender,
      TCloseAction &Action)
{
tsumm = 0; tproizv = 1;
Label11->Caption = "0";
}
```

```
//---------------------------------------------------------------------------
                                        «Sotrudniki.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Sotrudniki.h"
#include "DataModule.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MainForm.h"
#include "MedTovari.h"
#include "Postavwiki.h"
#include "Proizvoditeli.h"
#include "VidTovara.h"
#include "MestoHraneniya.h"
#include "Prihod.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpSotrudniki *SpSotrudniki;
//---------------------------------------------------------------------------
__fastcall TSpSotrudniki::TSpSotrudniki(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpSotrudniki::FormShow(TObject *Sender)
{
DM->DOLJNOSTI->Active = true;
DBGrid1->Columns->Items[3]->PickList->Clear();
DM->DOLJNOSTI->First();
while (!DM->DOLJNOSTI->Eof) {
DBGrid1->Columns->Items[3]->PickList->Add(DM->DOLJNOSTINAME-
>Value);
DM->DOLJNOSTI->Next();
}
}
//---------------------------------------------------------------------------
void __fastcall TSpSotrudniki::Button1Click(TObject *Sender)
{
DM->SOTR->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpSotrudniki::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->SOTR->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpSotrudniki::Button3Click(TObject *Sender)
{
if (DM->SOTR->Modified) DM->SOTR->Post();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TSpSotrudniki::DBGrid1TitleClick(TColumn *Column)
{
if (DM->SOTR->IndexName == "SOTR_SOTR") DM->SOTR->IndexName =
"UNQ1_SOTRUDNIKI";
else DM->SOTR->IndexName = "SOTR_SOTR";
}
//---------------------------------------------------------------------------
                                        «VidTovara.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "VidTovara.h"
#include "DataModule.h"
#include "MainForm.h"
#include "Edinici.h"
#include "MedTovari.h"
#include "Doljnosti.h"
#include "MestoHraneniya.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "ViruchkaMain.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TSpVidTovara *SpVidTovara;
//---------------------------------------------------------------------------
__fastcall TSpVidTovara::TSpVidTovara(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TSpVidTovara::Button1Click(TObject *Sender)
{
DM->VID->Insert();
DBGrid1->SetFocus();
}
//---------------------------------------------------------------------------
void __fastcall TSpVidTovara::Button2Click(TObject *Sender)
{
if (Application->MessageBox("Действительно хотите удалить
запись?","Подтвердите удаление записи",
MB_YESNO + MB_ICONEXCLAMATION) == IDYES)
DM->VID->Delete();
}
//---------------------------------------------------------------------------
void __fastcall TSpVidTovara::Button3Click(TObject *Sender)
{
if (DM->VID->Modified) DM->VID->Post();
}
//---------------------------------------------------------------------------
void __fastcall TSpVidTovara::DBGrid1TitleClick(TColumn *Column)
{
if (DM->VID->IndexName == "SORT_VID") DM->VID->IndexName =
"UNQ1_VID_TOVARA";
else DM->VID->IndexName = "SORT_VID";
}
```

```
//---------------------------------------------------------------------------
                                «Viruchka.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "ViruchkaMain.h"
#include "DataModule.h"
#include "MainForm.h"
#include "Doljnosti.h"
#include "Edinici.h"
#include "MedTovari.h"
#include "MestoHraneniya.h"
#include "Postavwiki.h"
#include "Prihod.h"
#include "Proizvoditeli.h"
#include "Rashod.h"
#include "ReportMedTovari.h"
#include "Sotrudniki.h"
#include "VidTovara.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TViruchka *Viruchka;
//---------------------------------------------------------------------------
__fastcall TViruchka::TViruchka(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TViruchka::FormShow(TObject *Sender)
{
Panel1->Caption = "";
Panel2->Caption = "";
Panel3->Caption = "";
DM->PribilZp->Close();
DM->PribilZp->SQL->Clear();
DM->PribilZp->SQL->Add("select VIRUCHKA, ZATRATI,  VIRUCHKA -
ZATRATI as prb from pribil");
DM->PribilZp->Open();
if (!DM->PribilZp->FieldByName("zatrati")->IsNull) Panel1->Caption = DM-
>PribilZp->FieldValues["zatrati"];
else Panel1->Caption ="0";
if (!DM->PribilZp->FieldByName("VIRUCHKA")->IsNull) Panel2->Caption =
DM->PribilZp->FieldValues["VIRUCHKA"];
else Panel2->Caption ="0";
if (!DM->PribilZp->FieldByName("prb")->IsNull) Panel3->Caption = DM-
>PribilZp->FieldValues["prb"];
else Panel3->Caption ="0";
}
//---------------------------------------------------------------------------
                                «Ystanovki.cpp»
//---------------------------------------------------------------------------
#include <vcl.h>
#pragma hdrstop
#include "Ystanovki.h"
#include "DataModule.h"
#include "MainForm.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
```

```
TNastroiki *Nastroiki;
//---------------------------------------------------------------------------
__fastcall TNastroiki::TNastroiki(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TNastroiki::rb1Click(TObject *Sender)
{
if (rb1->Checked) {
Edit1->Enabled = true;
Label1->Enabled = true;
Button2->Enabled = false;
Edit1->Text = "";
}
}
//---------------------------------------------------------------------------
void __fastcall TNastroiki::rb2Click(TObject *Sender)
{
if (rb2->Checked) {
Edit1->Enabled = false;
Label1->Enabled = false;
Button2->Enabled = true;
Edit1->Text = "localhost";
}
}
//---------------------------------------------------------------------------
void __fastcall TNastroiki::Button2Click(TObject *Sender)
{if (OpenDialog1->Execute()) {
Edit2->Text = OpenDialog1->FileName;}}
//---------------------------------------------------------------------------
void __fastcall TNastroiki::Button1Click(TObject *Sender)
{
DM->IBDatabase1->Params->Clear();
DM->IBDatabase1->LoginPrompt = false;
DM->IBDatabase1->DatabaseName = Edit1->Text + ":" + Edit2->Text;
DM->IBDatabase1->Params->Add("user_name=SYSDBA");
DM->IBDatabase1->Params->Add("password=masterkey");
DM->IBDatabase1->Params->Add("lc_ctype=win1251");
try {
DM->IBDatabase1->Connected = true;
DM->VID->Active = true;
DM->MED->Active = true;
DM->ED->Active = true;
DM->PROIZV->Active = true;
DM->SOTR->Active = true;
DM->POST->Active = true;
DM->PRIHOD->Active = true;
DM->RASHOD->Active = true;
DM->DOLJNOSTI->Active = true;
DM->MESTO->Active = true;
StatusBar1->Panels->Items[1]->Text = Edit2->Text;
ShowMessage("Успешно");
} catch (Exception &e)
{ShowMessage("Не удалось подключиться");}
}
//---------------------------------------------------------------------------
void __fastcall TNastroiki::FormCreate(TObject *Sender)
{rb2->Checked = true;}
```

ПРИЛОЖЕНИЕ Б

ИСХОДНЫЕ ДАННЫЕ

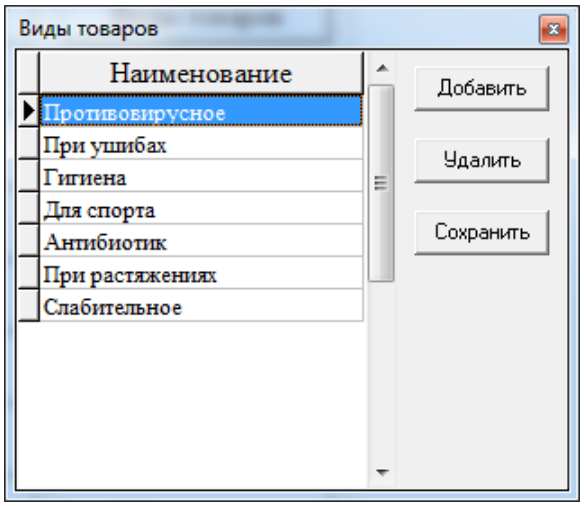

Рисунок Б.1 - Данные из справочника «Виды товаров»

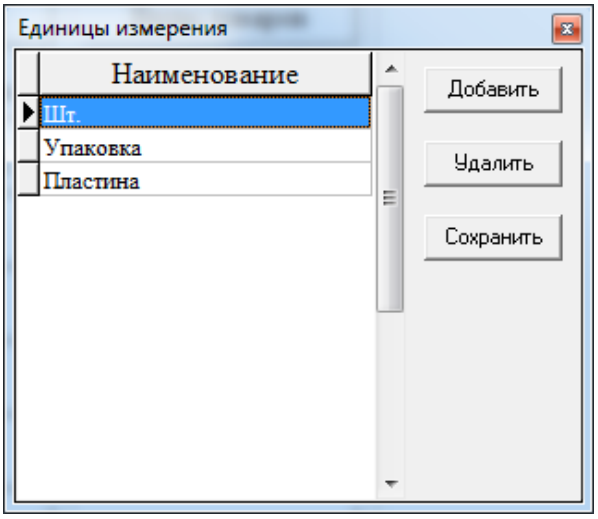

Рисунок Б.2 - Данные из справочника «Единицы измерения»

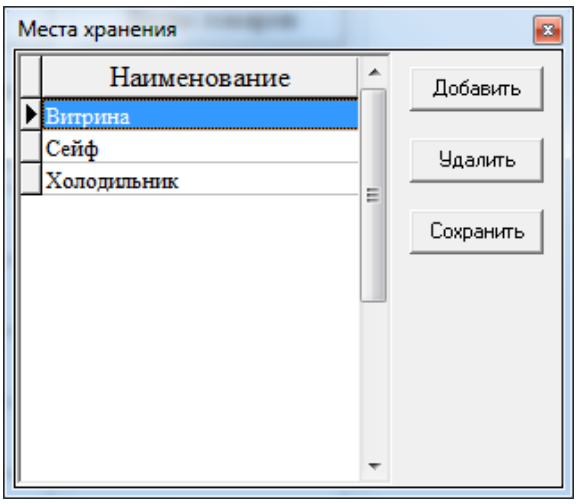

Рисунок Б.3 - Данные из справочника «Места хранения»

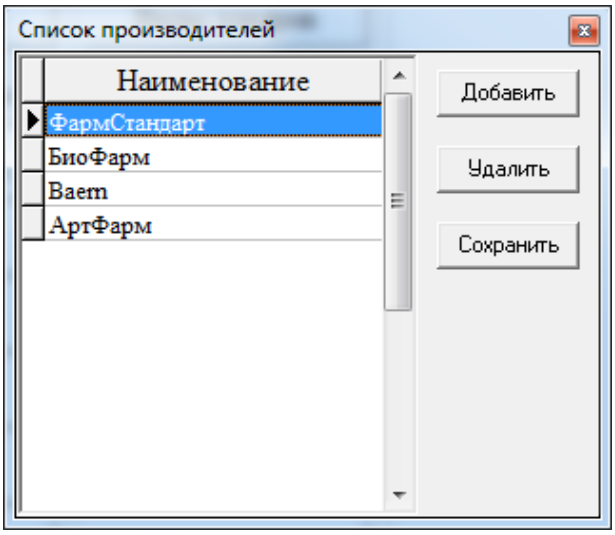

Рисунок Б.4 - Данные из справочника «Список производителей»

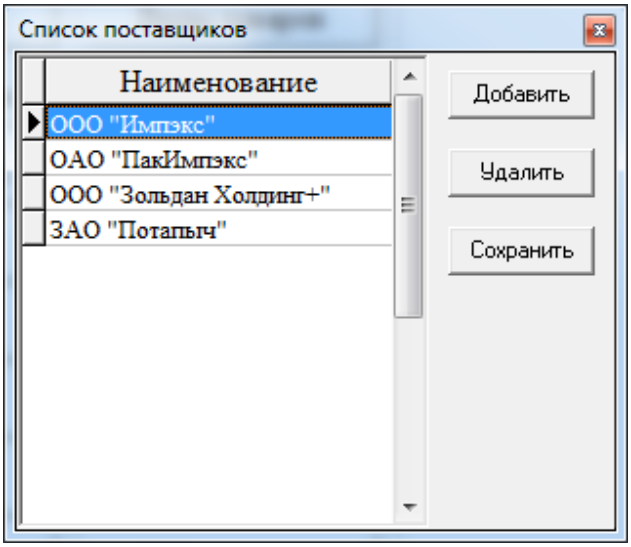

Рисунок Б.5- Данные из справочника «Список поставщиков»

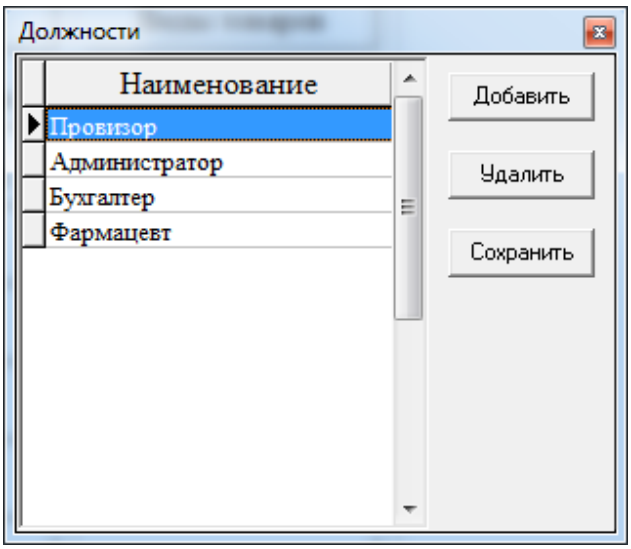

Рисунок Б.6 - Данные из справочника «Должности»

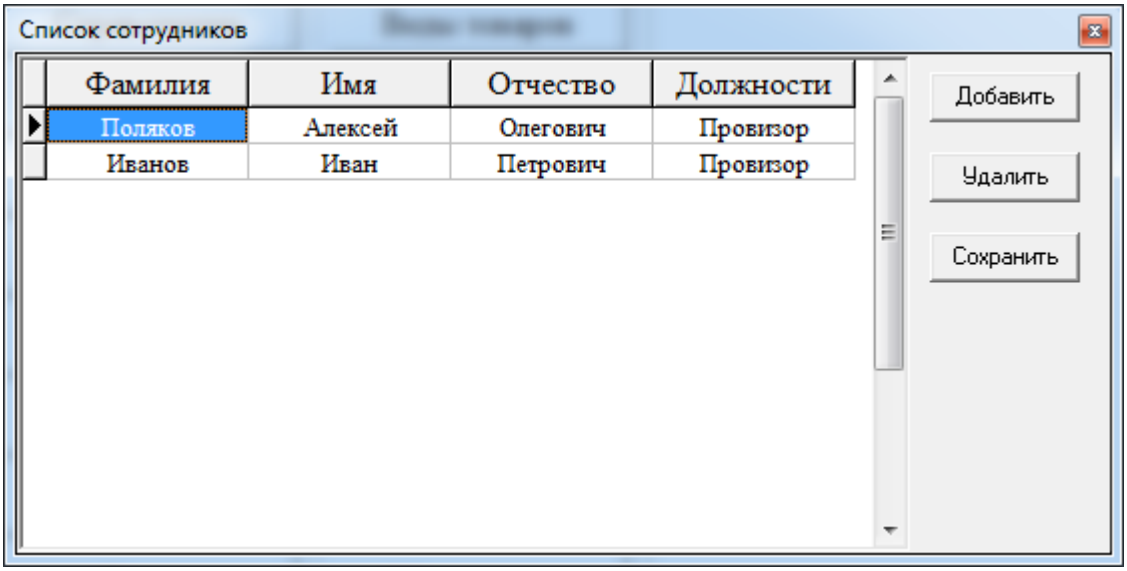

Рисунок Б.7 - Данные из справочника «Список сотрудников»

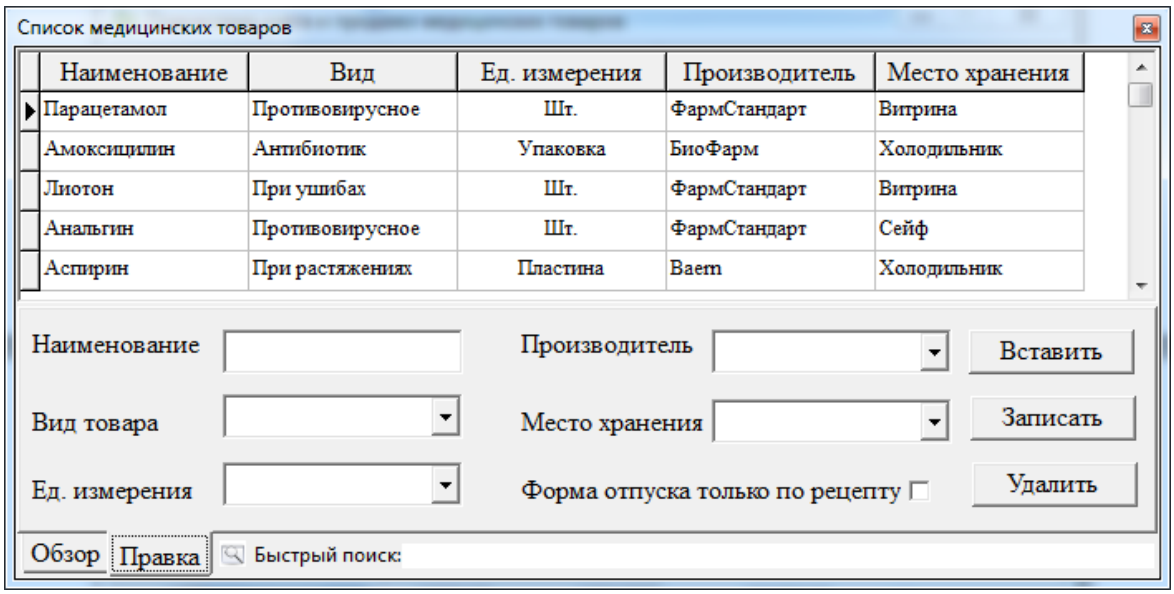

Рисунок Б.8 - Данные из справочника «Список медицинских товаров»

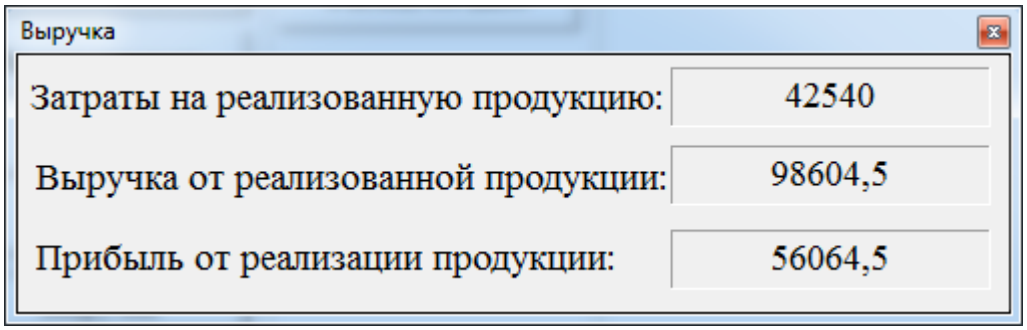

Рисунок Б.9 - Данные вкладки «Выручка»

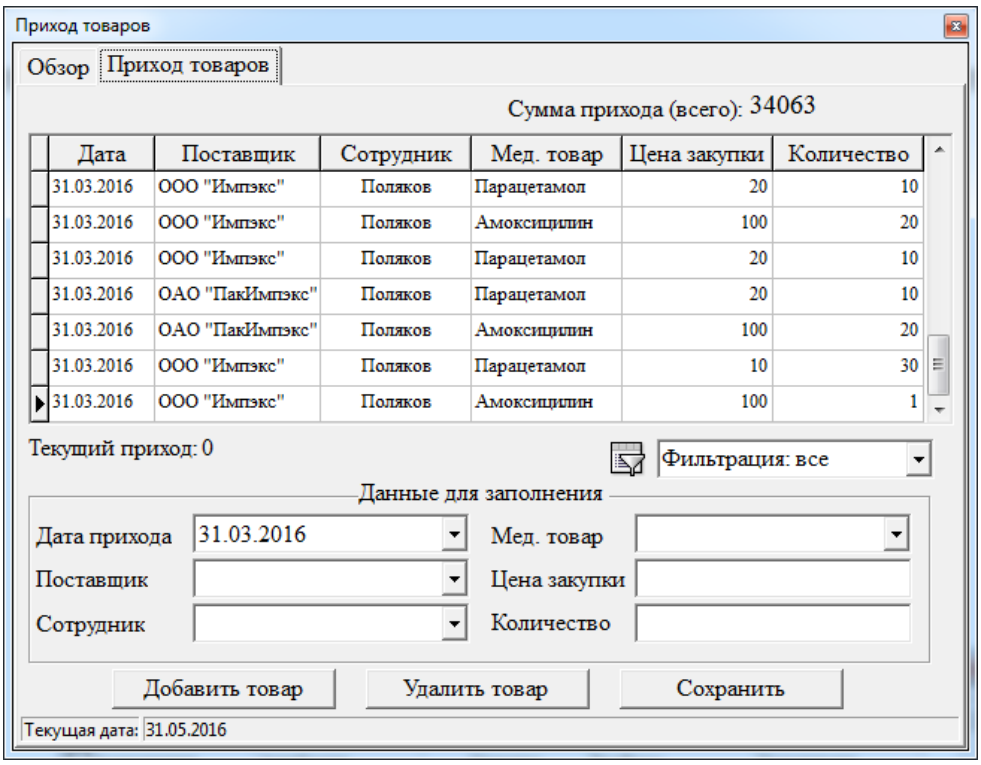

Рисунок Б.10 - Данные документа «Приход товаров»

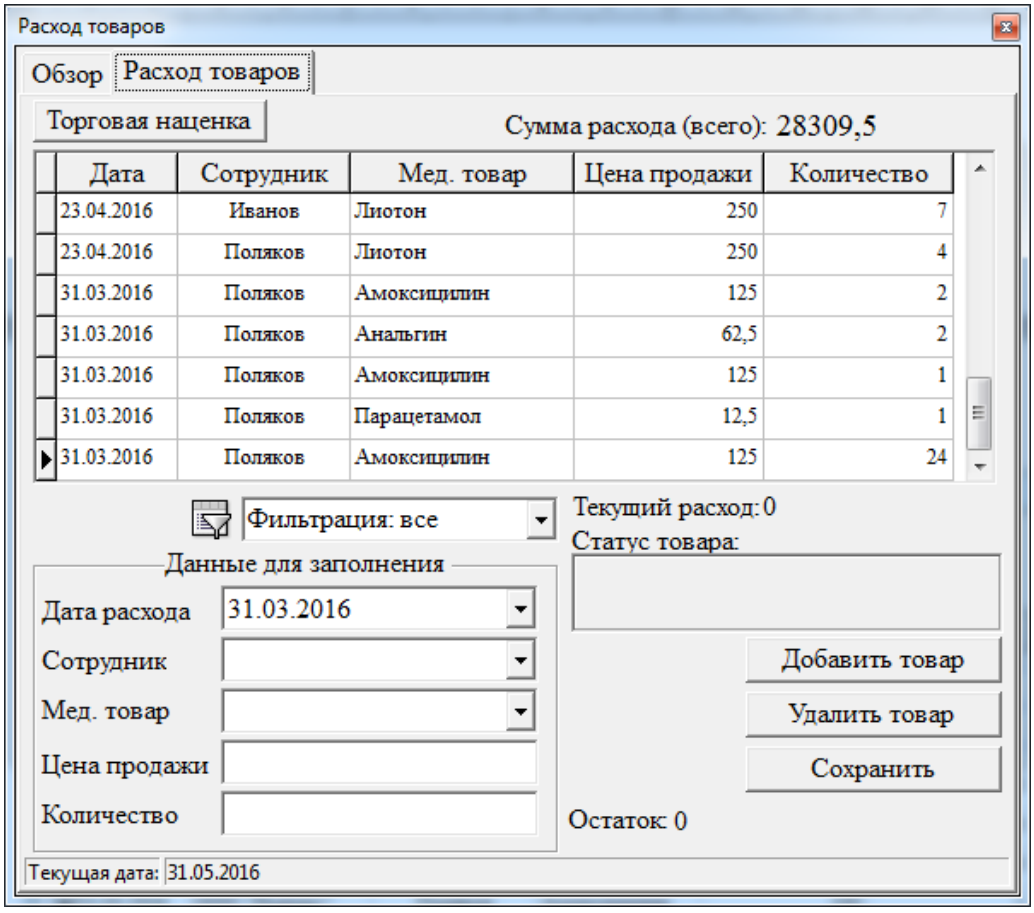

Рисунок Б.11 - Данные документа «Расход товаров»