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 ПОДСИСТЕМЫ УЧЕТА ЗАЯВОК НА ОБСЛУЖИВАНИЕ КОМПЬТЕРНОЙ ТЕХНИКИ ДЛЯ ЗАО «СОФТКОННЕКТ»

Выпускная квалификационная работа

студента заочной формы обучения
09.03.03 прикладная информатика
5 курса группы 07001151
Агаркова Алексея Станиславовича

Научный руководитель:
к.т.н.,доцент
Маматов Е.М.

БЕЛГОРОД 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Независимо от производственной сферы одну из главных ролей в предприятии играет оперативность получения и передачи нужных сведений об определенных сегментах производственного процесса. Эта характеристика напрямую зависит от технического обеспечения, которое позволяет автоматизировать и рационализировать процесс производства и обслуживания.

Переход к упорядоченному, автоматизированному ведению бизнеса очевиден, но не для всех без исключения предприятий. Многие управленцы считают автоматизацию процессом затруднительным, несвоевременным в период после кризиса.

Однако же, отечественный рынок информационных технологий не стоит на месте и в настоящее время существуют тиражные программы, которые способны обеспечить выполнение всех необходимых функций и решение важных задач управления проектами.

Для предприятий, которые уже долгое время занимаются внедрением технологий автоматизации производства, в настоящее время наиболее актуально оптимизация работы вспомогательной инфраструктуры, то есть различных подразделений, обслуживающих основное производство. В масштабах крупного предприятия реинжиниринг деятельности даже небольшого отдела может привести к значительному повышению эффективности работы предприятия в целом.

Общая цель ВКР это автоматизация деятельности отдела информационных технологий ЗАО «СофтКоннект», а именно разработка автоматизированной подсистемы учета заявок на обслуживание компьютерной техники. В ходе достижения общей цели ВКР решаются следующие задачи:

- изучение структуры и общих принципов деятельности работы компании;
- подробное изучение деятельности отдела информационных технологий ЗАО «СофтКоннект» и проведение анализа данной деятельности;
- обоснование необходимости автоматизации деятельности на основании результатов анализа;
- разработка приложения для автоматизации деятельности отдела информационных технологий по обслуживанию корпоративных пользователей.

ВКР состоит из четырех глав, которые посвящены отдельным вопросам. Первая глава работы освящает общую характеристику предметной области. В данной главе рассматривается структура корпорации в целом, и устанавливаются общие требования к будущему программному средству. Вторая и третья главы ВКР рассматривают проектные вопросы и вопросы, связанные с реализацией и тестированием проекта. Далее в последней главе приводится технико-экономический анализ рассматриваемого проекта.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Общая характеристика отдела информационных технологий

Отдел информационных технологий является одним из структурных подразделений ЗАО «СофтКоннект». Отдел информационных технологий взаимодействует с различными отделами предприятия по вопросам аппаратного и программного обеспечения компьютерной техники, используемой для непосредственной деятельности отделов. В связи с этим руководитель отдела несет ответственность за издержки, возникающие при простое отделов, связанном с неработоспособностью технических средств.

Основными задачами отдела информационных технологий ЗАО «СофтКоннект» являются:

- поддержание работоспособности корпоративной компьютерной сети и управление ее ресурсами;

- управление доступом корпоративных пользователей к ресурсам сети;

- поддержание работоспособности аппаратного обеспечения всех подразделений корпорации;

- администрирование корпоративного сайта, обновление представленной на нем информации, а также взаимодействие с частными лицами по средствам корпоративного сайта;

- поддержание работоспособности и актуальности программного обеспечения рабочих мест пользователей;

- анализ рынка аппаратных и программных средств, а также возможностей их внедрения для использования в корпорации;

- анализ деятельности предприятий в составе корпорации на предмет необходимости автоматизации данной деятельности;

- разработка специализированного программного обеспечения для автоматизации конкретных бизнес-процессов;

- внедрение и сопровождение приобретенных или разработанных

собственными силами информационных систем;

- консультирование пользователей по конкретным вопросам использования тех или иных программных или аппаратных средств.

1.2 Описание информационного обеспечения предприятия

Для реализации управления экономической деятельности предприятия активно используются информационные системы корпоративного использования. Чаще всего, это системы на базе платформы 1С: Предприятие, и комплексная система управления реализации продукции L-Express. Основу этих систем составляет клиент-серверная архитектура. Функционирование подобных систем поддерживается аппаратными ресурсами серверных компьютеров на опредприятии.

Комплекс программного обеспечения «1С:Предприятие 8» состоит из платформы и прикладных решений, которые были реализованы разработанные на ее основе, для поддержки деятельности организаций и частных лиц автоматизированным способом. Платформа же не является программным продуктом для использования конечными пользователями, они зачастую работают с небольшой частью многих прикладных решений (конфигураций), которые разрабатываются на этой платформе. Реализация такого подхода даёт возможность автоматизировать многие виды деятельности, при одновременном использовании единой технологической платформыц.

Возможность перестроения элементов платформы позволяет извлекать пользу из использования 1С:Предприятия 8 в разнообразных областях деятельности:

1. Процессы автоматизации производственных предприятий и предприятий торговли, финансовых организаций и бюджетных, а так же предприятий занимающихся сферой обслуживания и т.д.

2. Реализация процесса поддержки оперативного управления

предприятием;

3. Помощь в автоматизации процессов управления организационной и хозяйственной деятельности;

4. Организация и сопровождение бухгалтерского учета с несколькими вариантами счетов и произвольными мерами учета, регламентированное составление отчетных документов;

5. управленческий учет с широкими возможностями и построения аналитической отчетности, поддержка учета с различными валютами;

6. реализация задач построения планирования, организации бюджета и анализа финансового положения;

7. вычисления заработной платы и управление кадрами;

другие области применения.

Наблюдение за внедрением прикладных решений на основе 1С: Предприятие 8 позволяет сделать вывод, что система может решать задачи различной степени сложности - от реализации процессов автоматизации одного рабочего места до создания систем информационного сопровождения в масштабах организации в целом.

Одновременно с этим, введение в эксплуатацию такой информационной системы повышает требования по сравнению с небольшим или средним внедрением. Системы информационного сопровождения в масштабах предприятия должна позволять реализовывать достаточную производительность при условии одновременной и интенсивной работы большого количества пользователей, ведь они будут использовать одни и те же информационные и аппаратные ресурсы.

Конфигурация 1С: Бухгалтерия

Конфигурация "Бухгалтерия предприятия" позволяет автоматизировать бухгалтерский и налоговый учет, в том числе подготовку обязательной (регламентированной) отчетности на предприятии. Бухгалтерский и налоговый учет проводится в соответствии с действующим законодательством Российской Федерации.

"1С:Бухгалтерия 8" поддерживает решение всех задач бухгалтерского отдела организации, если отдел бухгалтерии полностью отвечает за учет на в организации, включая, например, выписку первичных документов, учет реализованной продукции и т. д. Данное программное обеспечение можно использовать исключительно для ведения учета бухгалтерской и налоговой документации.

В состав конфигурации включен план счетов бухгалтерского учета, соответствующий Приказу Минфина РФ "Об утверждении плана счетов бухгалтерского учета финансово-хозяйственной деятельности организаций и инструкции по его применению" от 31 октября 2000 г. № 94н (в редакции Приказа Минфина РФ от 07.05.2003 № 38н). Состав счетов, организация аналитического, валютного, количественного учета на счетах соответствуют требованиям законодательства по ведению бухгалтерского учета и отражению данных в отчетности. При необходимости пользователи могут самостоятельно создавать дополнительные субсчета и разрезы аналитического учета.

Основным способом отражения хозяйственных процессов в проведении учета является возможность ввода документов программы, которые являются первичными бухгалтерскими документами. Так же, допускается непосредственный ввод отдельных проводок. Для группового ввода проводок можно использовать типовые операции – простой инструмент автоматизации, легко и быстро настраиваемый пользователем.

Конфигурация 1С: Управление торговлей

"1С:Управление торговлей 8" — это актуальный пакет ПО которое позволяет повысить эффективность торговых предприятий.

"1С:Управление торговлей 8" позволяет комплексно провести автоматизацию решения задач оперативного и управленческого учета, анализа и планирования торговых операций, что в свою очередь позитивно скажется на эффективности принятия управлених решений в рамках торгового предприятия.

Область применения автоматизации реализуемая с помощью "1С:Управление торговлей 8" можно представить следующим образом:

1. управление отношениями с клиентами организации;
2. управление реализацией продукции (включая оптовую, розничную и комиссионную торговлю);
3. управление закупкой;
4. анализ и управление ценовой политикой;
5. управление запасами на складах;
6. управление финансами;
7. учет коммерческих затрат;
8. учет НДС;
9. мониторинг и анализ эффективности торговой деятельности.

Комплексная система управления отгрузкой продукции L-Express

Основной целью проекта L-Express является предоставление комплексных решений по автоматизации технологических процессов на бетонных заводах, производствах сухих смесей, других предприятиях строительной индустрии.

Такой подход позволяет охватить в рамках единого автоматизированного технологического комплекса все взаимосвязанные с основным технологическим процессом службы. На текущий момент система L-Express позволяют построить единый комплекс автоматизированных систем, включающих:

1. АСУ ТП основного производства (растворы, бетонные и сухие строительные смеси или другая продукция), отвечающую за управление основным технологическим процессом производства
2. АСУ склада инертных материалов
3. АСУ склада цемента
4. АСУ адресной раздачи бетона
5. АСУ термовлажностной обработки железобетонных изделий
6. АСУ диспетчерской службы, отвечающую за процессы

оперативного управления обслуживанием заказов клиентов и оформление первичной документации

7. АСУ лаборатории, отвечающую за контроль качества продукции, ввод в технологическую базу данных результатов анализов и корректировку дозировочных составов на основании этих анализов

8. АСУ весового хозяйства, отвечающую за контроль количества поступающего на предприятие сырья, а также ввод первичных документов на поступившее сырье

9. Систему контроля доступа (СКД) на территорию предприятия, выполняющую функции контроля прохода сотрудников, посетителей и автотранспорта на территорию предприятия, а также контроля документов на вывоз отгруженной продукции

10. Комплексное решение для предприятий по производству ЖБИ, включающее в своем составе практически все вышеперечисленные системы управления и позволяющее охватить в рамках единого автоматизированного технологического комплекса практически всю технологическую цепочку производства ЖБИ, начиная от складов сырья (цемента, инертных материалов, добавок), управления бетоносмесительными узлами, адресной раздачей бетона и заканчивая термовлажностной обработкой изделий, их складирования и контроля вывоза с территории предприятия.

11. АРМы управленческого персонала предприятия, являющиеся клиентами единой технологической базы данных и отвечающие за контроль исполнения производственных планов, работы оборудования основного производства, а также формирование сводных отчетов о работе производства

12. Модули для интеграции технологической базы данных с системами управления производством (АСУП) на базе 1С, Галактики и других систем.

Таким образом, в отличие от большинства конкурентов, L-Express предлагает единый комплекс взаимодействующих между собой в масштабе реального времени автоматизированных систем с единой технологической

базой данных. Благодаря этому, вся информация, необходимая включенным в систему службам, становится доступной им сразу же после ее занесения в базу данных, что существенно сокращает затраты времени оперативного и управленческого персонала на согласование действий и выработку оперативных и управленческих решений. Естественно, что на объекте заказчика может устанавливаться не весь описанный набор систем, а только необходимые ему модули.

Модульный принцип построения системы L-Express позволяет сконфигурировать ее с учетом реальных потребностей предприятия, а также позволяет с успехом использовать систему L-Express на предприятиях с разными технологическими процессами (бетонные заводы, цементные заводы и терминалы, карьеры, элеваторы и комбикормовые заводы, мукомольные заводы и др.), где используется весодозирующее оборудование. Для адаптации системы к тем или иным условиям производства достаточно подобрать необходимый набор модулей и выполнить ряд параметрических настроек.

Система L-Express является многоуровневой распределенной системой управления технологическим процессом производства и отгрузки продукции потребителям.

На нижнем уровне системы управления находятся промышленные контроллеры, управляющие технологическим процессом.

Оператор осуществляет лишь общий контроль над ходом технологического процесса со своего рабочего места, оборудованного персональным компьютером, на котором установлена программа. Весь технологический процесс представлен в удобной форме в виде мнемосхемы со всеми основными параметрами контроля и управления. В штатном режиме вмешательство оператора в технологический процесс не требуется. В случае возникновения нештатных ситуаций программа управляющего контроллера переведет технологический процесс в безопасное состояние, диагностирует ситуацию и предоставит оператору принять то или иное решение.

Рабочие места операторов неразрывно связаны с остальными подсистемами, занимающимся контролем качества, планированием и диспетчеризацией отгрузок, контролем доступа на территорию предприятия, а также обменом с системой учета предприятия.

В основе программного обеспечения верхнего уровня системы L-Express лежит технология клиент-сервер, которая позволяет достичь высокой производительности обработки данных, ее надежного хранения и оперативного получения необходимой информации о работе предприятия. На данный момент система работает под управлением SQL-сервера FireBird версии 2.

Большим достоинством системы является сочетание простоты использования и широких функциональных возможностей. Пользователи с невысокой квалификацией могут достаточно легко освоить все операции, необходимые им для выполнения своих должностных обязанностей. С другой стороны квалифицированные пользователи по достоинству оценят функциональные возможности по обработке информации в системе. В систему встроено большое количество различных фильтров, позволяющих пользователям просматривать существующую информацию в различных разрезах, имеются средства по экспорту данных в файлы форматов MS Word, MS Excel, HTML, PDF, а также пересылке отчетов по электронной почте. Благодаря своей архитектуре и характеристикам программа легко интегрируется с системами автоматизации учета на предприятии (1С, Галактика и.т.п.).

1.3 Анализ деятельности отдела информационных технологий «как есть»

Основной задачей отдела информационных технологий является обслуживание корпоративных пользователей. Данную деятельность можно представить в виде диаграмм IDEF0 (рисунок 2).

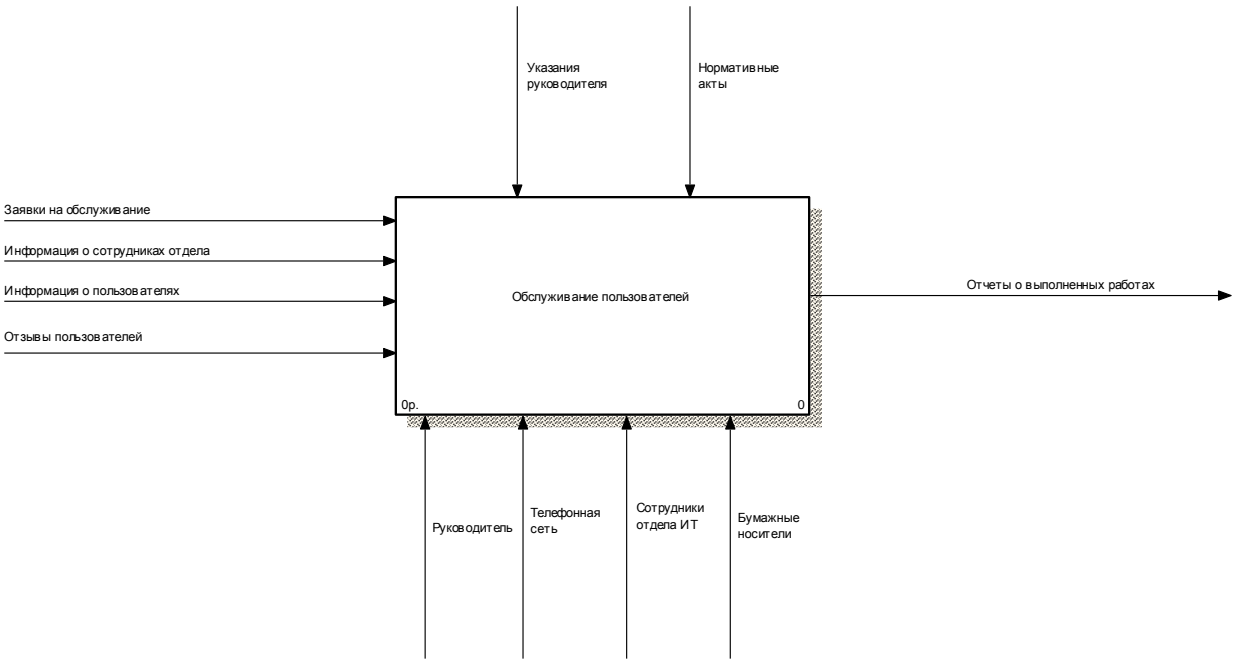

Рисунок 2 - Контекстная диаграмма «Обслуживание пользователей»

На данной диаграмме присутствуют такие входные данные как заявки на обслуживание, информация о пользователях, информация о сотрудниках отдела ИТ, а также отзывы пользователей о проделанной работе. В процессе обслуживания пользователей принимают участие как сотрудники отдела, так и сам руководитель, при этом для регистрации заявок используется внутренняя телефонная сеть предприятия а также бумажные носители. Более подробно данный процесс представлен на диаграмме декомпозиции на рисунке 3.

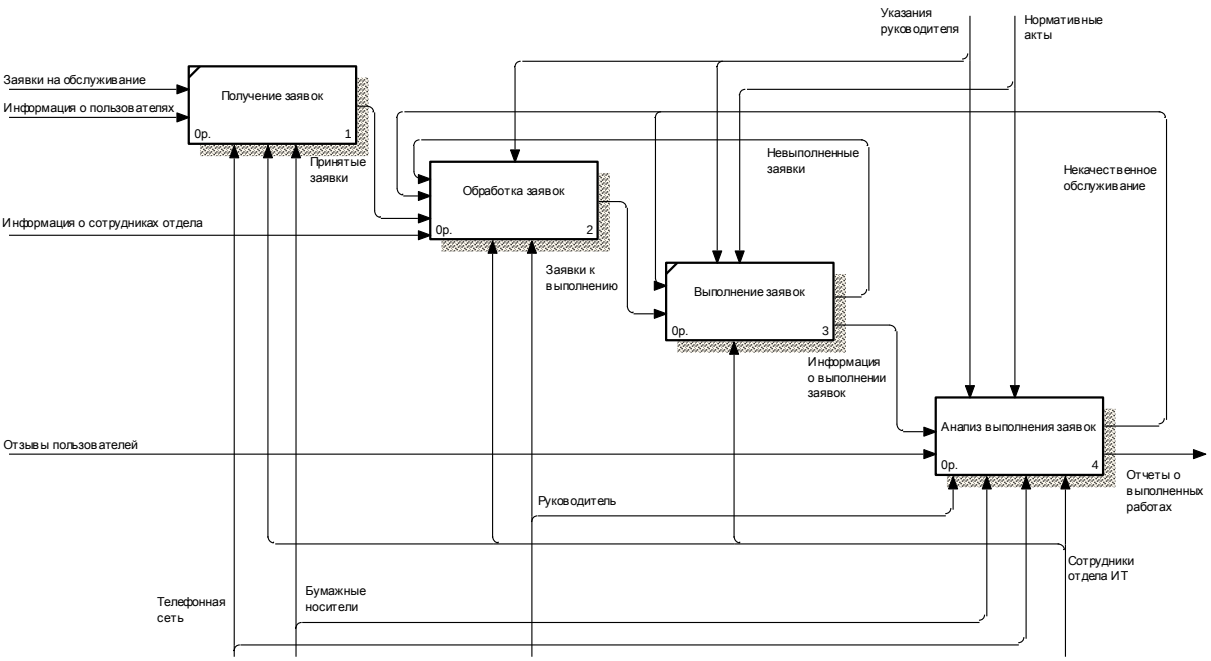

Рисунок 3 - Диаграмма декомпозиции «Обслуживание пользователей»

Как видно из диаграммы процесс обслуживания начинается с получения сотрудниками отдела заявки по телефонной сети и ее регистрации на бумажных носителях. Затем сотрудники отдела вместе с руководителем производят первичную обработку заявки с целью определения ее важности, срочности выполнения, а также необходимых материалов, временных затрат, требований к квалификации сотрудника выполняющего заявку, после чего заявка передается данному сотруднику на выполнение. В случае невозможности ее выполнения, по каким бы то ни было причинам, заявка возвращается на обработку. Информация о выполненных заявках также анализируется с целью составления отчетов о деятельности отдела и определения качества обслуживания. Здесь также используются бумажные носители. Далее обратимся к процессу обработки заявок, представленному на рисунке 4.

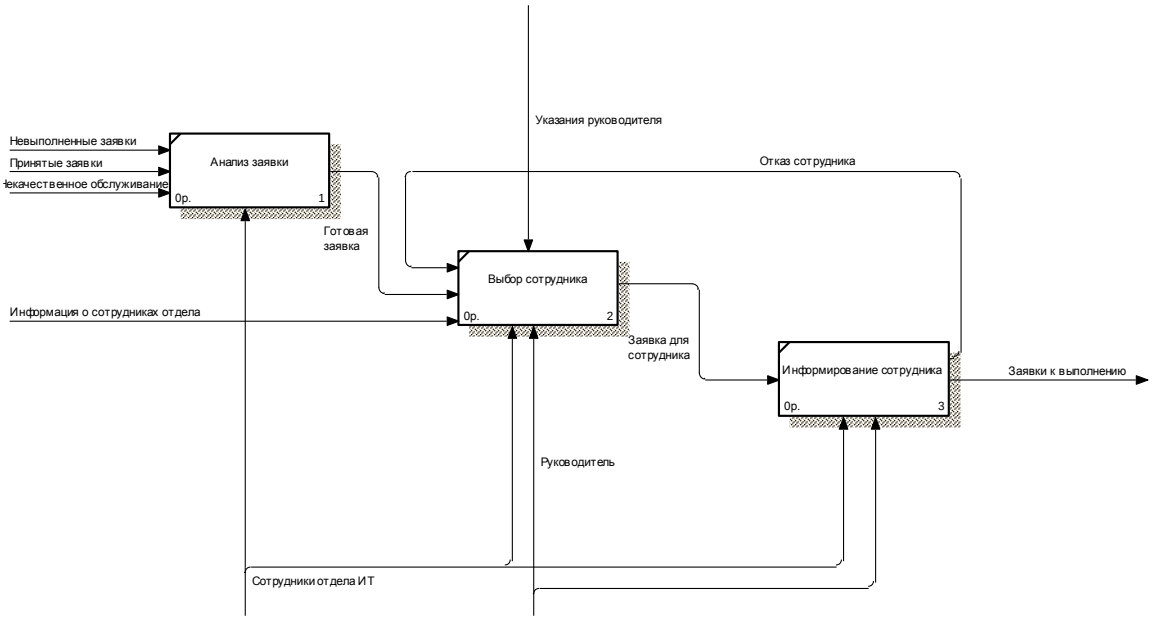

Рисунок 4 - Диаграмма декомпозиции «Обработка заявок»

На этапе анализа заявки происходит определение основных причин возникновения проблемы и возможные способы ее устранения. Далее руководитель и сотрудники отдела определяют сотрудника для обслуживания заявки в соответствии с его квалификацией. На завершающем этапе анализа сотруднику направляется заявка на выполнение.

Далее необходимо рассмотреть процесс анализа выполненных заявок (рисунок 5).

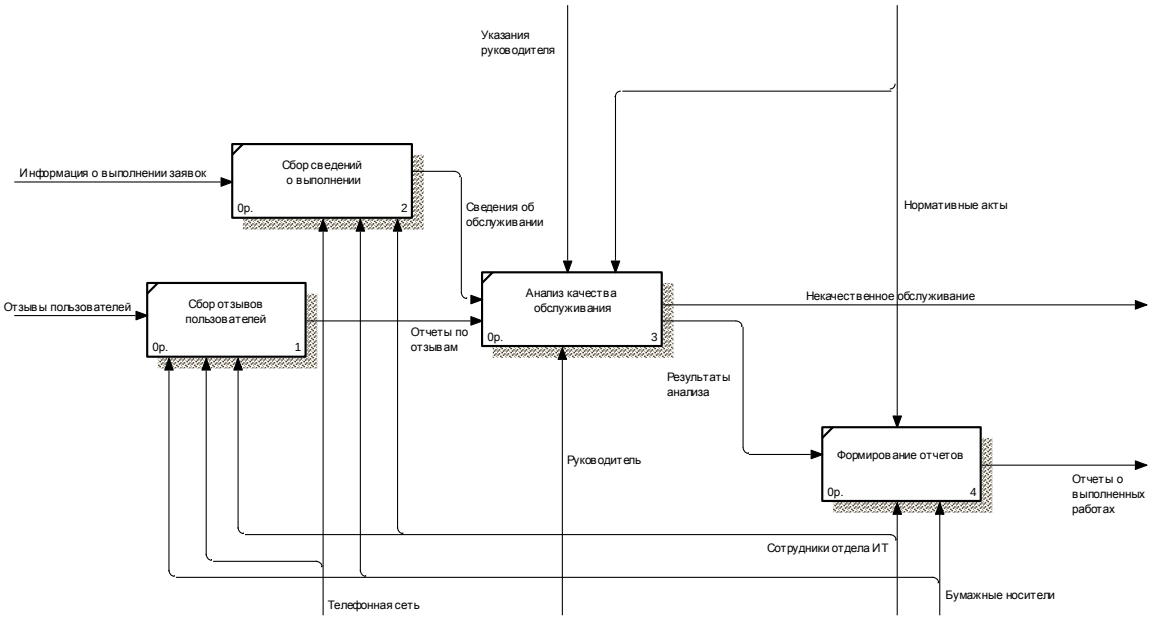

Рисунок 5 - Диаграмма декомпозиции «Анализ выполненных заявок»

На представленной диаграмме видно, что процессы осуществляются вручную сотрудниками отдела с использованием бумажных носителей. В процессе сбора информации и выполненных заявках и отзывов пользователей активно применяется внутрифирменная телефонная сеть, а также средства сотовой связи. Анализ деятельности производится руководителем отдела на основе данных собранных на бумажных носителях. Формированием отчетов занимаются сотрудники отдела также вручную.

1.4 Требования к разрабатываемой подсистеме

1.4.1 Функциональные требования к подсистеме

Данная подсистема предназначена для использования сотрудниками отдела информационных технологий ЗАО «СофтКоннект» в процессе выполнения своих обязанностей при оформлении и сопровождении заявок на обслуживание компьютерной техники, а также корпоративными пользователями АРМ в процессе подачи и контроля заявок.

Целью данной автоматизированной подсистемы является повышение оперативности, производительности и уровня организации труда сотрудников отдела информационных технологий ЗАО «СофтКоннект», упрощение процесса подачи заявок конечными пользователями, а также централизованное хранение и обработка данных о заявках и пользователях корпорации.

Для реализации поставленных целей система должна отвечать следующим функциональным требованиям:

- автоматизированный контроль процесса оформления заявки, контроль ошибок;

- централизация данных в едином хранилище - базе данных;

- автоматизация процесса обработки заявок;

- удобство редактирования вспомогательной информации;
- разграничение доступа пользователей и администраторов;
- взаимодействие интерфейсов с базой данных системы.

1.4.2 Требования к надежности

Надежность подсистемы должна обеспечиваться на уровне используемых аппаратных и программных средств. Необходимый уровень надежности достигается путем контроля входных данных, разграничением прав доступа к хранилищу данных, централизованным хранением и обработкой данных на сервере.

1.4.3 Требования к информационной и программной совместимости

Автоматизированная подсистема должна обеспечивать информационную совместимость с известными приложениями операционной системы Windows (Word, Excel, Access). Программная совместимость обеспечивается автоматически в связи с использованием программных средств, совместимость которых обеспечена конструктивно (на этапе их создания) – Borland C++ Builder, IBExpert и т.д. Система реализуется под операционной системой Windows и СУБД Firebird 2.1.

1.4.4 Требования к техническому и программному обеспечению

Клиентская модуль подсистемы ориентирован на использование на персональных компьютерах класса IBM PC, начиная с Pentium II,

включенного в локальную сеть, объем оперативной памяти 32 Мб и свободного места на жестком диске 2Мб.

Программные требования: Windows NT 4.0 (Service Pack 5), 2000, XP.

Серверный модуль подсистемы обладает большими требованиями в первую очередь к объему оперативной памяти и жесткого диска. Рекомендуемые требования: объем оперативной памяти не менее 256 Мб и 2 Гб свободного места на жестком диске.

Основное программное требование серверной части подсистемы является наличие FireBird server 2.1.

2 ОБОСНОВАНИЕ ПРОЕКТНЫХ РЕШЕНИЙ

2.1 Выбор методологии проектирования

Основу проекта любой информационной системы составляют методологии, технологии и инструментальные средства проектирования. Методология можно реализовать через конкретные технологии и поддерживающие их стандарты, методики и инструментальные средства. Они обеспечивают выполнение процессов проектирования. Следовательно, очень важно правильно выбрать методологию разработки.

Стоит отметить, что на данный момент реализованно множество различных методологий, но всё большую популярность получила **методология XP (Extreme Programming)**. В ней можно выделить 4 составляющих:

- Коммуникация;
- Обратная связь;
- Простота;
- Смелость.

Из данных компонент следуют 12 практик, которым должны придерживаться проекты, использующие XP. Большинство этих практик представляют собой старые проверенные техники. Эти техники, тем не менее, многие успели забыть (включая большинство предсказуемых процессов). Методология XP не только возвращает к жизни такие техники, но и объединяет их таким образом, что все они поддерживают и усиливают друг друга. Тестированию отводиться определяющая роль в методологии XP. Тестирование является той базой, на которой строится разработка программного продукта. Следует отметить, что каждый программист пишет тесты одновременно с кодом разрабатываемой системы. Данные тесты используются при постоянной интеграции и в процессе сборки системы, что

помогает построить стабильный фундамент для дальнейшей работы. Используя данный фундамент методология XP строит эволюционный процесс проектирования, который основан на реорганизации кода системы в течение каждой последующей итерации.

Microsoft Solution Framework (MSF). Модель процесса в методологии MSF представляет собой общую методологию разработки и внедрения ИТ-решения. Следует отметить, что важное отличие этой модели заключается в том, что благодаря своей гибкости и отсутствию жёстко навязываемых процедур она может применяться при разработке весьма обширного круга ИТ-проектов. Жизненный цикл данного проекта MSF делит на пять фаз:

- обследование;
- планирование;
- разработка;
- тестирование;
- внедрение.

На каждом из уровней процесса формирования решения MSF предполагает цикличность: формирование версии продукта – цикл итераций, итерация – цикл из ежедневно собираемых билдов, билд – цикл изменений, которые вносяться в систему контроля версий.

Фазы и вехи процесса разработки :

- концепция проекта;
- планы проекта утверждены;
- разработка завершена;
- готовность решения;
- внедрение завершено.

Также методология MSF предполагает управление рисками выбранного проекта (риск – всякое событие или условие, оказывающее либо позитивное, либо негативное влияние на проект).

RUP (Rational Unified Process) – это программный продукт, который был разработан компанией Rational Software, который на данный момент в

наибольшей степени соответствует стандартам и нормативным документам, связанным с процессами жизненного цикла программного обеспечения и оценкой технологической зрелости организации разработчиков (ISO 12207, ISO 9000).

Следует отметить одну из отличительных особенностей технологии RUP. Таковой является принцип итерационного и инкрементного подхода к созданию программного обеспечения. Следовательно, в соответствии с этим принципом, вся разработка системы осуществляется в несколько «отрезков» – небольших проектов длительностью от 2 до 6 недель. Так вот эти краткосрочные проекты называются итерациями. Каждая из этих итераций включает свои собственные уровни анализа требований, проектирования, реализации, тестирования, интеграции и оканчивается формированием рабочей системы. Итерационный цикл базируется на постоянном увеличении и дополнении системы с периодической обратной связью и приспособлении дополняемых модулей к ядру системы.

Согласно технологии RUP, весь процесс жизненного цикла программного обеспечения делиться на отдельные циклы, которые, в свою очередь, также разбиваются на 4 стадии каждый:

Начальная стадия – исследование интересующей предметной области и разработка бизнес-плана проекта;

Стадия разработки (уточнение плана) – выполняется высокоуровневый анализ данной предметной области и создание проекта для построения базовой архитектуры системы, избавляются от наиболее рискованных элементов проекта;

Стадия конструирования – используется инкрементный подход, то есть каждая стадия итерации вносит очередные новые конструкции к вариантам использования, и программный код получает возможность частично переписываться для того, чтобы сделать его более приспособленным;

Следующая стадия ввод в действие (переход) – осуществляется перенос готового продукта в пользование заказчика (производиться тест

системы, разбираются ошибки, проходят обучение пользователи и специалисты, организовывается сопровождение).

В рамках RUP определяют шесть дисциплин:

- деловое моделирование;
- требования;
- анализ и проектирование;
- выполнение;
- испытание;
- развертывание.

А также три вспомогательные дисциплины:

- управление конфигурацией и изменениями,
- управление проектом;
- среда.

Дисциплина соответствует понятию технологического процесса и представляет собой последовательность действий, приводящих к получению значимого результата.

Также одной из отличительных особенностей данной технологии RUP является принцип планирования и управления проектом на основе функциональных требований к системе – вариантов использования. Целью варианта использования является то, что для того чтобы определить законченный аспект или фрагмент поведения некой сущности без раскрытия внутренней структуры этой сущности. В качестве такой сущности может выступать основная система либо любой другой фрагмент модели, который обладает собственным поведением, как в подсистемах или классах в моделях системы.

Отметим преимущества данной методологии:

- универсальность – RUP можно использовать в различных проектах;

- использование итерационного цикла;

- объектный подход – основывается на понятиях объектов, классов и взаимосвязей между ними.

Существующая возможность понижения стоимости проекта и сроков разработки при помощи правильного применения принципов планирования и управления проектом.

Планирование и управление проектом – ведется на основе функциональных требований к системе – вариантов использования.

Снижение рисков – все наиболее рискованные фрагменты проекта удаляются, выделяются все наиболее возможные риски и методы борьбы с ними.

Из перечня рассмотренных выше методологий предстоит выбрать именно ту, которая будет наиболее подходящей для осуществления проектирования. Следует принять во внимание специфику разрабатываемой автоматизированной подсистемы. В нашем случае удобнее будет применять метод балльных оценок для выбора.

Список выбранных критериев следующий: универсальность, использование итерационного цикла, объектный подход, планирование и управление проектом и ведение их на основе вариантов использования, снижение рисков. Критерии субъективны и выбраны по важности для разработчика.

После того как были рассмотрены все технологии и определения важности критериев, по пятибалльной шкале выставляются баллы по критериям отбора. Сумма произведений важности и баллов является необходимой итоговой оценкой. Произведя необходимые расчеты, становится ясно, что наибольшее количество баллов набрала методология RUP. Поэтому следует отдать предпочтение методологии RUP.

2.2 Обоснование проектных решений по техническому обеспечению

Для эффективного решения поставленной задачи необходимо соответствующее техническое обеспечение. Техническое обеспечение

данного проекта включает в себя непосредственно ЭВМ (системный блок), монитор, клавиатуру, манипулятор типа мышь.

При выборе ЭВМ необходимо руководствоваться рядом характеристик. К таким характеристикам относятся надежность, стоимость, производительность, объем памяти и другие.

От значения указанных параметров зависит возможность работы с требуемыми программными средствами, а следовательно, и успех создания системы.

В настоящее время в мире существуют ЭВМ нескольких классов: большие, мини - и микро-ЭВМ. Большие ЭВМ имеют очень высокую стоимость и быстродействие и предназначены для решения сложных задач, требующих большого количества вычислений. Они применяются при проведении фундаментальных научных исследований, в космической отрасли, в ядерной физике и т.д. Типичным представителем класса микро-ЭВМ являются персональные ЭВМ (ПЭВМ). Мини-ЭВМ занимают промежуточное место между большими и микро-ЭВМ.

Для решения экономических задач наиболее подходят ПЭВМ. Они имеют невысокую стоимость, небольшие размеры (умещаются на части стола) и подходящие характеристики быстродействия, надежности, объема памяти. Таким образом, они могут применяться практически на любом предприятии и, в частности, в юридических службах.

При выборе ПЭВМ для реализации комплекса поставленных задач учитываются такие характеристики:

- скорость обработки информации (тактовая частота процессора);
- объем оперативной памяти; этот фактор также влияет на скорость обработки информации;
- объем жесткого диска, который влияет на возможности хранения данных;
- наличие периферийных устройств;
- другие технические характеристики ПЭВМ.

2.3 Обоснование проектных решений по информационному обеспечению (ИО)

Понятие информационного обеспечения возникло с созданием автоматизированных систем управления (АСУ). Информационное обеспечение состоит из внутримашинного, которое включает массивы данных (входные, промежуточные, выходные), программы для решения задач, и внемашинного, которое включает системы классификации и кодирования оперативных документов, нормативно-справочной информации (НСИ).

Одно из важных требований к информационному обеспечению - это достоверность данных информационной базы.

Необходимая достоверность данных в информационных базах обеспечивается высокой степенью контроля на всех стадиях работы с данными.

Особенности технологии обработки данных связаны с такими факторами, как: функционирование в режиме диалога с пользователем, наличие накопителей информации, исключение бумажных технологий для обработки информации.

В состав технологических операций входят:

- загрузка программы;
- ввод данных;
- контроль информации и возможность корректировки;
- справочно-информационное обслуживание;
- формирование информационных массивов;
- вывод информации.

Существует несколько способов регистрации первичной информации:

- документальный;
- документальный с регистрацией на машинном носителе;
- автоматический.

По способу установления связей между данными различают реляционную, иерархическую и сетевую модели. Реляционная модель является простейшей и наиболее привычной формой представления данных в виде таблиц. Иерархическая и сетевая модели предполагают наличие связей между данными, имеющими какой-либо общий признак. В иерархической модели такие связи могут быть отражены в виде дерева-графа, в сетевой возможны связи “всех со всеми”.

В настоящее время реляционные системы лучше соответствуют техническим возможностям персональных компьютеров. Скоростные характеристики этих СУБД поддерживаются специальными средствами ускоренного доступа к информации - индексирование баз данных. Для создания подсистемы учета заявок можно предложить СУБД Firebird , которая располагает большим количеством мощных средств для работы с базами данных.

2.4 Обоснование проектных решений по программному обеспечению (ПО)

Программное обеспечение – совокупность программ системы обработки информации и программных документов, необходимых для эксплуатации этих программ (ГОСТ 19781-90)

Программное обеспечение – совокупность программ, процедур и правил, а также документации, относящихся к функционированию системы обработки данных.

Программное обеспечение можно классифицировать по нескольким признакам:

По назначению на:

- системное;
- прикладное;
- инструментальное.

По способу распространения и использования на:

- закрытое;
- открытое;
- свободное.

Свободное программное обеспечение может распространяться, устанавливаться и использоваться на любых компьютерах дома, в офисах, школах, вузах, а также коммерческих и государственных учреждениях без ограничений.

На данный момент на компьютерах корпорации «ЖБК-1» установлено лицензионное программное обеспечение таких фирм как Microsoft, 1С, Kaspersky Lab, Firebird, LExpress, ВЭД, Adobe и многих других.

Достоинства данного ПО следующие:

- широкая распространенность продукта;
- привычный интерфейс;
- стабильность работы;
- совместимость;
- большое количество разнообразных сред разработки;
- высокий уровень безопасности;
- широкие возможности настройки.

К недостаткам можно отнести:

- высокая ресурсоемкость;
- дороговизна;
- необходимость в постоянном обновлении.

2.5 Обоснование проектных решений по технологическому обеспечению

От того насколько рационально будет спроектирован технологический процесс, настолько гарантировано будет снижение стоимостных, трудовых затрат.

Технологический процесс, как правило, состоит из нескольких этапов. Целью первого этапа является сбор, регистрация, передача данных для дальнейшей обработки. Результатом обычно является составление документа. Цель второго этапа - перенос данных на машинные носители и первоначальное формирование информационной базы. Третий этап включает операции накопления, сортировки, корректировки и обработки данных. [7]

При выборе варианта технологического процесса требуется учитывать следующие требования:

- обеспечение достоверности обрабатываемой информации;
- решение задач в установленные сроки;
- обеспечение минимальных трудовых и стоимостных затрат на обработку данных;
- наличие возможности обработки данных на ЭВМ;
- возможность решения задачи в различных режимах.

Исходя из перечисленных выше требований целесообразно проектирование подсистемы, которое позволит децентрализовать процесс решения задачи и повысить производительность.

При обработке данных желательно использовать массивы нормативно-справочной информации. Это дает преимущества в скорости поиска, выбора, сортировки и т.д. При этом необходима возможность просмотра полученных результатов перед оформлением и передачей выходной информации. Очень актуальным становится вопрос выбора режима: пакетный или диалоговый.

Пакетный режим позволяет уменьшить вмешательство пользователя в процесс решения задачи и требует от него только выполнения операций по вводу и корректировке данных, но вместе с этим появляется вероятность полной загрузки ЭВМ, что не всегда удобно для пользователя.

Практика показывает, что использование подсистем с применением методов построения модели на основе диалога обеспечивает более гибкую связь пользователя с ЭВМ.

Диалоговый режим имеет ряд преимуществ: удобен при работе с базой; обеспечение защиты при несанкционированном доступе; обеспечивает непосредственное участие пользователя в процессе решения задачи; управляемость процессом; быстрый доступ, поиск и выдача информации в любой момент времени, выбор различных режимов работы; осуществление быстрого перехода от одной операции к другой.

Существует несколько типов диалога: управляющие команды, запросы, меню, диалог на ограниченном естественном языке.

2.6 Выбор средств разработки программного обеспечения

Прежде чем приступать к разработке, нужно выбрать программные продукты, которые будут применяться в ходе построения системы. Сегодня, когда сложность программных проектов постоянно возрастает, повышаются требования к эффективности их реализации. Данные исследований в области разработки программного обеспечения говорят о том, что результаты как минимум половины «внутренних» проектов разработки программных средств не оправдывают возложенных на них надежд. В этих условиях становится особенно актуальной задача оптимизации всего процесса создания программных средств с охватом всех его участников - проектировщиков, разработчиков, тестеров, служб сопровождения и менеджеров. Управление жизненным циклом приложений (Application Lifecycle Management, ALM) рассматривает процесс выпуска программных средств как постоянно повторяющийся цикл взаимосвязанных этапов:

- определение требований (Requirements);
- проектирование и анализ (Design & Analysis);
- разработка (Development);
- тестирование (Testing);
- развертывание и сопровождение (Deployment & Operations).

Каждый из этих этапов должен тщательно отслеживаться и контролироваться. Правильно организованная ALM-система позволяет:

- сократить сроки вывода продуктов на рынок (разработчикам приходится заботиться лишь о соответствии их программ сформулированным требованиям);

- повысить качество, гарантируя при этом, что приложение будет соответствовать потребностям и ожиданиям пользователей;

- повысить производительность труда (разработчики получают возможность делиться передовым опытом разработки и внедрения);

- ускорить разработку благодаря интеграции инструментов;

- уменьшить затраты на сопровождение, постоянно поддерживая соответствие между приложением и его проектной документацией;

- получить максимальную отдачу от инвестиций в навыки, процессы и технологии.

Особенность процесса разработки состоит в том, что, с одной стороны, это процесс итерационный, с другой стороны, он не всегда последовательно переходит от одного этапа к другому. Нередко от тестирования происходит возврат к проектированию, анализу или ещё более ранним этапам. Кроме этого, необходимо заметить, что внутренняя организация процесса может значительно модифицироваться в зависимости от корпоративных регламентов и особенностей конкретных проектов.

Реализация ALM-стратегии в исполнении Borland заключается в предоставлении комплекса взаимосвязанных инструментов для всех этапов жизненного цикла приложений. В таблице 1 представлены продукты семейства систем Borland ALM и их назначение. [4]

Таблица 1 - Семейство систем Borland ALM, дополняющее .NET Framework

Название продукта	**Назначение**
Borland CaliberRM	Согласование требований
Borland Together	Проектирование программ с помощью UML
Borland C#Builder	Разработка на C# с частичной поддержкой VB.NET
Borland Delphi	Разработка программ на Delphi/Pascal
Borland OptimizeIt Profiler	Профилирование программ
Borland Janeva	Обеспечение совместимости с Enterprise JavaBeans, J2EE и CORBA
Borland InterBase	Реализация встраиваемых баз данных, используемых совместно с Windows Server 2003
Borland StarTeam	Управление: конфигурирование и координация изменений

На рисунке 6 изображены стадии жизненного цикла программного обеспечения и программные продукты Borland, которые могут быть использованы на этих этапах.

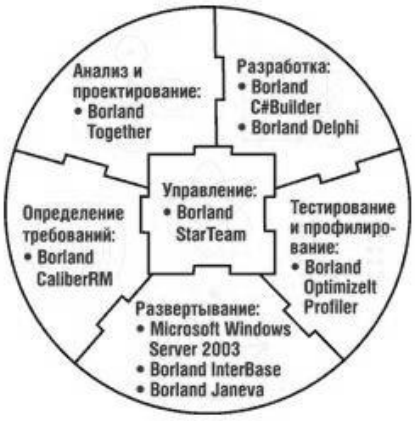

Рисунок 6 - Стадии жизненного цикла программного обеспечения Borland

Для анализа компании возможно использование AllFusion Process Modeler 7 (BPwin) — мощное средство системного анализа деловой и производственной активности от компании, позволяющее адекватно отслеживать соответствие структуры бизнеса, документооборота, финансовых потоков жестким и динамичным требованиям экономики. Система BPwin поможет повысить конкурентоспособность, оптимизировать процессы управления. Результатом использования BPwin является

исключение лишних и бесполезных действий, снижение затрат, повышение гибкости и эффективности всего вашего бизнеса. BPwin — незаменимый инструмент менеджеров и бизнес-аналитиков, а в руках системных аналитиков и разработчиков — еще и мощное средство моделирования процессов при создании корпоративных информационных систем.

Функциональные возможности AllFusion Process Modeler 7 (BPwin):

- поддержка нескольких нотаций (IDEF0 (рекомендации Госстандарта РФ, федеральный стандарт США), потоков работ IDEF3 (федеральный стандарт США) и потоков данных (DFD);
- интуитивно-понятный графический интерфейс;
- анализ показателей затрат и производительности;
- свойства, определяемые пользователем (UDP);
- организационные графики;
- методы контроля корректности модели;
- интерфейс к средствам имитационного моделирования;
- документальный центр проекта;
- работа с моделями бизнес-процессов из собственных программных приложений;
- интеграция процессов/данных;
- собственный генератор шаблонов отчетов. Report Template Builder.

Недостатки BPwin:

- нельзя менять мышью размеры текстового поля. Приходится при наборе текста делать перевод строки так, чтобы размеры были такие, какие требуется на диаграмме;
- отсутствует возможность самому редактировать текст Header/footer;
- английские слова в заголовках формы заменить на русские невозможно;
- несколько дубоватый, не совсем Windows-ный интерфейс: при сохранении, при закрытии проекта. Одно «Close without saving» чего стоит;

- устаревшие кириллические шрифты, которые, если диаграммы копировать, не видны в MS Office 2000, XP;

- невозможность выделения и копирования одного или группы объектов, - только диаграмма полностью.

Примеры эффективного использования AllFusion Process Modeler 7 (BPwin):

- административные органы: NATO;
- разработка КИС: НИИ "Восход";
- финансовые структуры: слияние двух бирж, автоматизация банков;
- страхование: StanCorp;
- недвижимость, строительство: Portman;
- правоохранительные органы: использование в криминалистике;
- армия, оборона: NATO, Quantum Research (англ.).

Для создания базы данных возможно использование утилиты IBExpert — GUI-оболочка, предназначенная для разработки и администрирования баз данных Interbase и Firebird т.е. реляционная система управления базами данных .

Как основные достоинства IBExpert разработчики указывают:

- поддержка Interbase версий 4.x, 5.x, 6.x, 7.x; Firebird 1.x, 2.x; Yaffil 1.x;
- работа одновременно с несколькими базами данных;
- отдельные редакторы для всех объектов БД с синтаксической подсветкой;
- мощный SQL редактор с историей запросов и возможностью фонового выполнения запросов;
- отладчик хранимых процедур и триггеров;
- поиск в метаданных;
- полное и частичное извлечение данных и метаданных;
- анализатор зависимостей объектов баз данных;
- отчеты по метаданным;

- менеджеры пользователей и пользовательских привилегий;
- экспорт данных в различные форматы.

IBExpert обладает множеством облегчающих работу компонентов: визуальный редактор для всех объектов базы данных, редактор SQL и исполнитель скриптов, отладчик для хранимых процедур и триггеров, построитель области, собственный скриптовый язык, а также дизайнер баз данных и т. д.

3 ПРОЕКТИРОВАНИЕ ПОДСИСТЕМЫ УЧЕТА И ОБРАБОТКИ ЗАЯВОК НА ЗАКУПКУ ОБОРУДОВАНИЯ И ТОВАРНО-МАТЕРИАЛЬНЫХ ЦЕННОСТЕЙ

3.1 Информационное обеспечение задачи «Учета заявок на обслуживание компьютерной техники»

3.1.1 Информационная модель и ее описание

Информационная модель — модель объекта, которая представленная в виде информации, описывающей существенные для данного рассмотрения параметры и переменные величины объекта, связи между ними, входы и выходы объекта и позволяющая путём подачи на модель информации об изменениях входных величин моделировать возможные состояния объекта. Информационные модели нельзя потрогать или увидеть, они не имеют материального воплощения, потому что строятся только на информации. Информационная модель — совокупность информации, характеризующая существенные свойства и состояния объекта, процесса, явления, а также взаимосвязь с внешним миром.

Информационные модели делятся на описательные и формальные.

Описательные информационные модели - это модели, созданные на естественном языке (т.е. на любом языке общения между людьми: английском, русском, китайском, мальтийском и т.п.) в устной или письменной форме.

Формальные информационные модели - это модели, созданные на формальном языке (т.е. научном, профессиональном или специализированном). Примеры формальных моделей: все виды формул, таблицы, графы, карты, схемы и т.д.

Программный продукт BPwin относится к мощным редакторам для создания моделей, которые позволяют вести анализ, документировать и планировать все изменения бизнес-процессов. [2]

В BPwin имеется возможность сбора всей интересующей информации о работе предприятия и графического отображения этой информации в виде целостной и непротиворечивой модели. Целостность и непротиворечивость проекта гарантируются рядом методологий и нотаций. В BPwin используются три таких методологии: IDEF0, DFD и IDEF3.

Также BPwin имеет возможность проверять созданные модели с точки зрения синтаксиса выбранной человеком методологии, проверяет ссылочную целостность между всеми диаграммами проекта, а также выполняет ряд других проверок. Однако сохраняются основные преимущества рисунка – простота создания и наглядность.[3]

BPwin реализует выполнение следующих функций:

- поддерживает сразу три стандартные нотации – IDEF0 (функциональное моделирование), DFD (моделирование потоков данных) и IDEF3 (моделирование потоков работ). Эти три основных ракурса позволяют описывать предметную область более комплексно;
- Дает возможность повысить эффективность бизнеса, оптимизирует любые процедуры в компании;
- Целиком и полностью поддерживает вариантый расчета себестоимости по объему хозяйственной деятельности (функционально-стоимостной анализ, ABC);
- позволяет упростить процесс сертификации на соответствие стандартам качества ISO9000;
- является стандартом де-факто, интегрирован вместе с программным продуктом ERwin (для моделирования БД), Paradigm Plus (для моделирования компонентов ПО);

- благодаря вышеупомянутой интеграции и поддержке совместной, командной работы над одними и теми же моделями (с помощью ModelMart;
- интегрирован со средством имитационного моделирования Arena;
- включает в себя собственный генератор отчётов;
- Эффективная манипуляция моделями – сливать и расщеплять их - одно из важнейших качеств данного продукта;
- имеет огромный набор средств документирования моделей, проектов.

Далее рассмотрим анализ деятельности отдела информационных технологий ЗАО «СофтКоннект» «как должно быть». Контекстная диаграмма деятельности отдела информационных технологий представлена на рисунке 7.

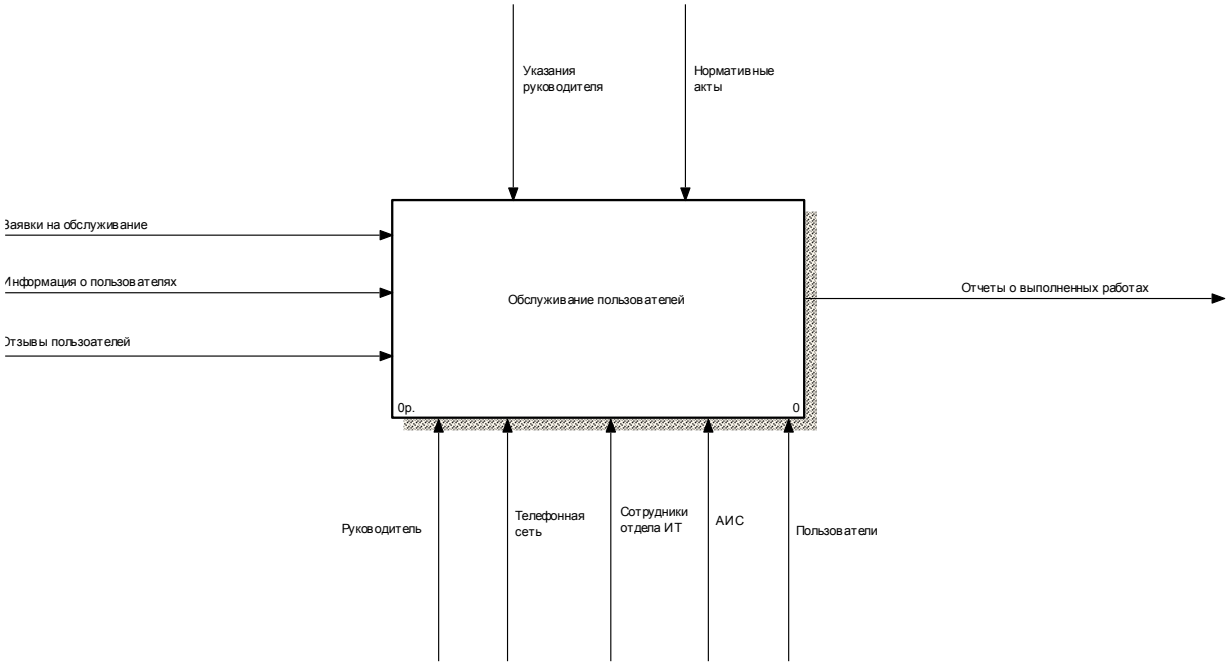

Рисунок 7 - Контекстная диаграмма «Обслуживание пользователей»

На данной диаграмме присутствуют такие входные данные как заявки на обслуживание, информация о пользователях и отзывы пользователей. В обслуживании участвуют сотрудники отдела ИТ, руководитель отдела, а также сами пользователи при помощи АИС. Рассмотрим диаграмму декомпозиции на рисунке 8.

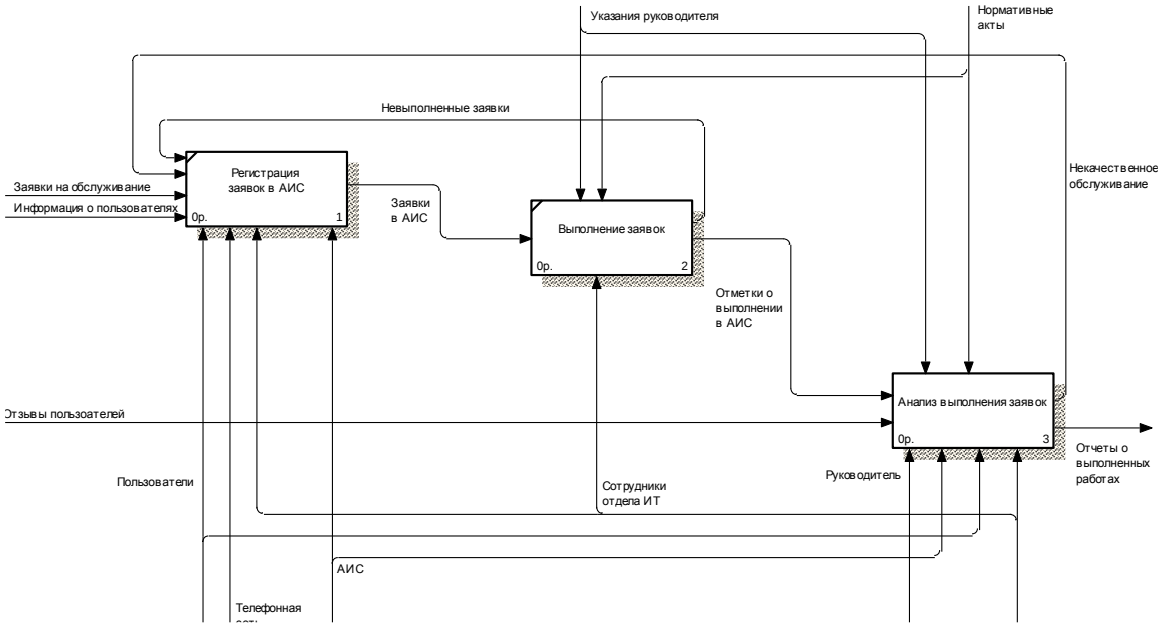

Рисунок 8 - Диаграмма декомпозиции «Обслуживание пользователей»

В отличие от анализа «как есть» в данной диаграмме отсутствует блок обработки заявок. Данную функцию полностью берет на себя АИС и освобождает от этой функции руководителя и сотрудников отдела ИТ. Также отличительной чертой является отсутствие использования не надежных бумажных носителей. Вместо этого используется АИС со встроенной БД для хранения информации. Теперь рассмотрим диаграмму процесса анализа выполненных заявок (рисунок 9).

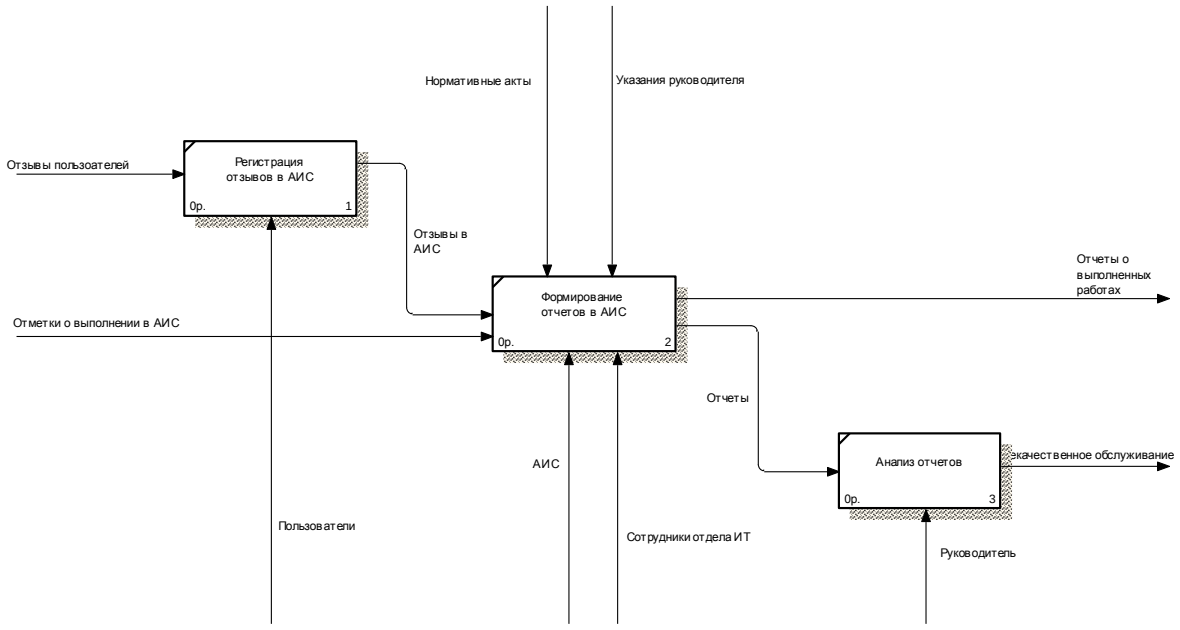

Рисунок 9 - Диаграмма декомпозиции «Анализ выполненных заявок»

На данной диаграмме также очевидны преимущества использования АИС в автоматизированном формировании отчетов, а также упрощении их анализа руководителем. В данном процессе также теперь не требуется использование бумажных носителей информации.

3.1.2 Характеристика базы данных

3.1.2.1 Характеристика инфологической модели БД

В результате моделирования системы удалось выделить основные объекты системы и их взаимосвязи, что позволяет перейти к проектированию структуры базы данных.

Под информационной базой понимается определенным образом организованная совокупность данных, хранимых в памяти системы в виде файлов, с помощью которых удовлетворяется информационные потребности управленческих процессов и решаемых задач. Существует несколько способов организации информационной базы. В данном проекте выбран способ создания реляционной базы данных.

Необходимо создать базу данных, в которой решались бы следующие задачи:

- ввод, хранение и поиск необходимой информации;
- ведение учета и отслеживание результатов работы.

Основным структурным компонентом базы данных является таблица, содержащая записи определенного вида и формы. Каждая запись таблицы содержит всю необходимую информацию об отдельном элементе базы данных. Такие отдельные структурные элементы называют полями таблицы.

Первым этапом при создании таблицы является определение перечня полей, из которых она должна состоять, их типов и размеров. При этом каждому полю присваивается уникальное имя.

Перед началом создания базы данных, ее нужно нормализовать. Нормализация базы данных – это процесс уменьшения избыточности информации в базе данных. Процесс проектирования базы данных с использованием метода нормальных форм заключается в последовательном переводе по определенным правилам отношений из первой нормальной формы в нормальные формы более высокого порядка.

Первый нормальная форма требует, чтобы каждое поле таблицы базы данных было не делимым, не содержало повторяющихся групп. Не делимость поля означает, что каждое поле не должно делится на несколько полей. А повторяющиеся группы указывают на то, чтобы поля не содержали одинаковые по смыслу значения.

Вторая нормальная форма требует:

1) все поля таблицы зависят от первичного ключа, то есть первичный ключ однозначно определен и является не избыточным;

2) те поля, которые зависят от части первичного ключа, должны быть выделены в отдельные таблицы.

Первичный ключ может быть простым (одно поле) или составным (несколько полей) – единственным в каждой таблице. Он обеспечивает:

- однозначную идентификацию записи в таблице;
- ускорение выполнения запросов к базе данных;
- установление связей между таблицами;
- создание ограниченной ссылочной целостности таблиц.

В процессе проектирования БД была сформирована ее модель средствами AllFusion Data Modeler. Данная модель представлена на рисунках 10 и 11.

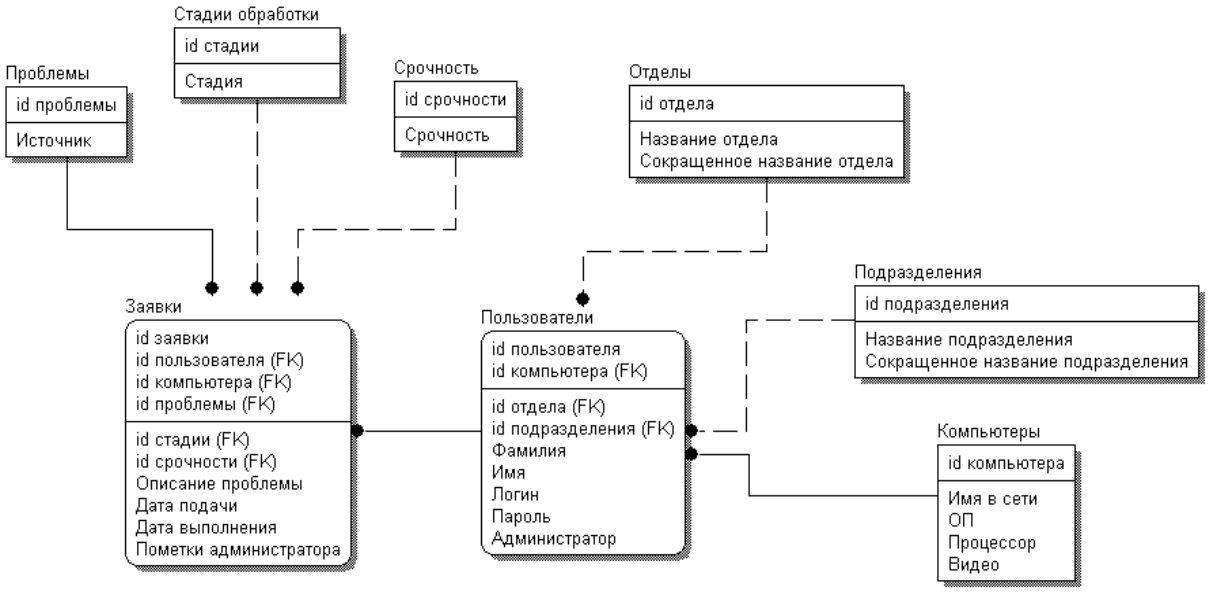

Рисунок 10 - Логическая модель

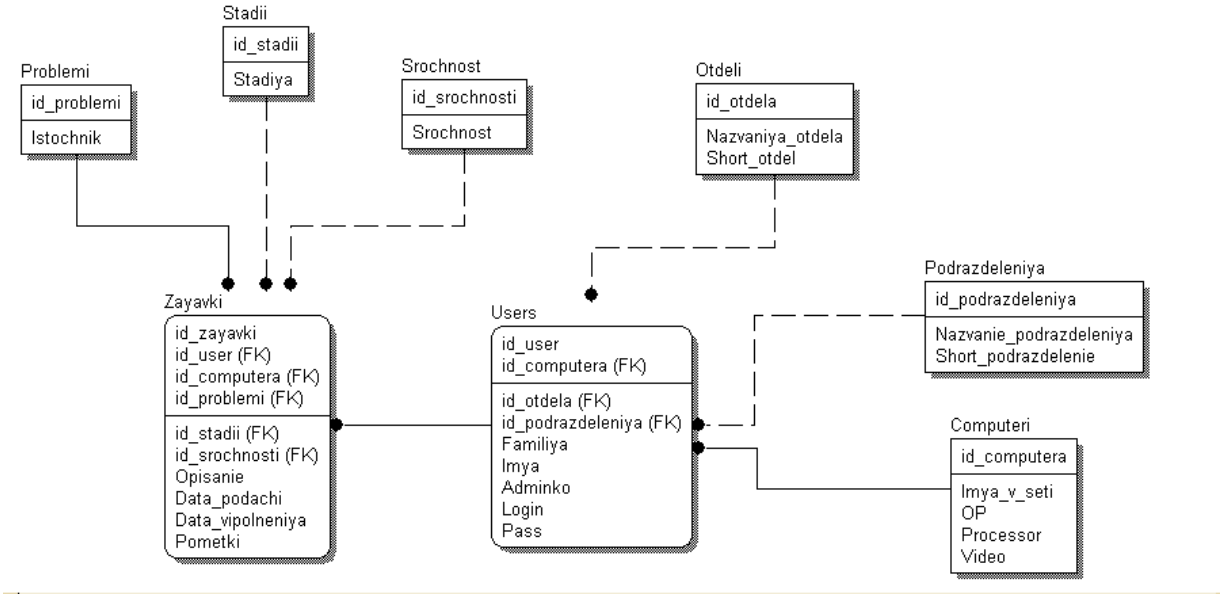

Рисунок 11 - Физическая модель

На данных моделях отражены 6 вспомогательных таблиц, напоминающих списки, и 2 главные сводные таблицы, в которых и происходят основные процессы регистрации заявок от пользователей.

3.1.2.2 Характеристика даталогической модели БД

После проведения анализа инфологической модели можно приступить к созданию даталогической - модели, отражающей логические взаимосвязи между элементами данных безотносительно их содержания и физической организации. При этом даталогическая модель разрабатывается с учетом конкретной реализации СУБД, также с учетом специфики конкретной предметной области на основе ее инфологической модели.

Даталогическую модель удобно представить в виде таблицы 2.

Таблица 2 – Даталогическая модель базы данных

Имя таблицы	Поле	Тип	Not Null	Primary key	Имя домена
PROBLEMI	ID_PROBLEMI	INTEGER	Да	Да	ID_DOMAIN
	ISTOCHNIK	CHAR(100)	Да		TEXT_DOMAIN
STADII	ID_STADII	INTEGER	Да	Да	ID_DOMAIN
	STADIYA	CHAR(100)	Да		TEXT_DOMAIN
SROCHNOST	ID_SROCHNOSTI	INTEGER	Да	Да	ID_DOMAIN
	SROCHNOST	CHAR(100)	Да		TEXT_DOMAIN
OTDELI	ID_OTDELA	INTEGER	Да	Да	ID_DOMAIN
	NAZVANIYA_OTDELA	CHAR(100)	Да		TEXT_DOMAIN
	SHORT_OTDEL	CHAR(15)	Да		SHORT_TEXT_DOMAIN
PODRAZDELENIYA	ID_PODRAZDELENIYA	INTEGER	Да	Да	ID_DOMAIN
	NAZVANIYA_ PODRAZDELENIYA	CHAR(100)	Да		TEXT_DOMAIN
	SHORT_ PODRAZDELENIE	CHAR(15)	Да		SHORT_TEXT_DOMAIN
COMPUTERI	ID_COMPUTERA	INTEGER	Да	Да	ID_DOMAIN
	IMYA_V_SETI	CHAR(100)	Да		TEXT_DOMAIN
	OP	CHAR(100)	Да		TEXT_DOMAIN
	PROCESSOR	CHAR(100)	Да		TEXT_DOMAIN
	VIDEO	CHAR(100)	Да		TEXT_DOMAIN
USERS	ID_USER	INTEGER	Да	Да	ID_DOMAIN
	ID_COMPUTERA	INTEGER	Да		ID_DOMAIN
	ID_PODRAZDELENIYA	INTEGER	Да		ID_DOMAIN
	ID_OTDELA	INTEGER	Да		ID_DOMAIN
	FAMILIYA	CHAR(100)	Да		TEXT_DOMAIN
	IMYA	CHAR(100)	Да		TEXT_DOMAIN
	ADMINKO	CHAR(15)	Да		SHORT_TEXT_DOMAIN
	LOGIN	CHAR(15)	Да		SHORT_TEXT_DOMAIN
	PASS	CHAR(15)	Да		SHORT_TEXT_DOMAIN
ZAYAVKI	ID_ZAYAVKI	INTEGER	Да	Да	ID_DOMAIN
	ID_USER	INTEGER	Да		ID_DOMAIN
	ID_PROBLEMI	INTEGER	Да		ID_DOMAIN
	ID_SROCHNOSTI	INTEGER	Да		ID_DOMAIN
	ID_STADII	INTEGER	Да		ID_DOMAIN
	OPISANIE	CHAR(500)			LONG_TEXT_DOMAIN
	DATA_PODACHI	TIMESTAMP			DATA_VREMYA_DOMAIN
	DATA_VIPOLNENIYA	TIMESTAMP			DATA_VREMYA_DOMAIN
	POMETKI	CHAR(500)			LONG_TEXT_DOMAIN

3.2 Разработка базы данных

Первоначально при создании базы данных необходимо создать домены, которые в дальнейшем будут использованы в полях таблиц. Создадим их с помощью стандартных средств IBExpert. Результат представлен на рисунке 12.

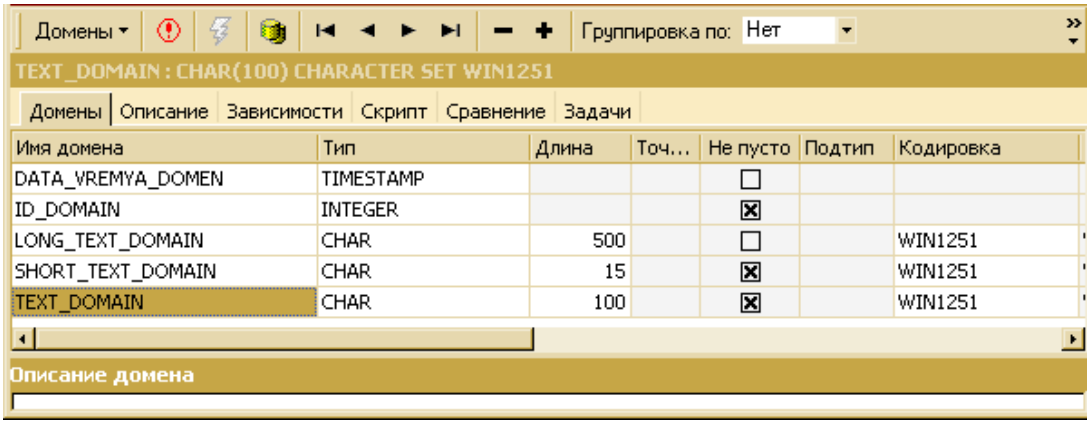

Рисунок 12 – Домены базы данных

Далее при помощи конструктора создаем все таблицы базы данных с указанием первычных и внешних ключей. На рисунке 13 приведен пример создания таблицы «USERS».

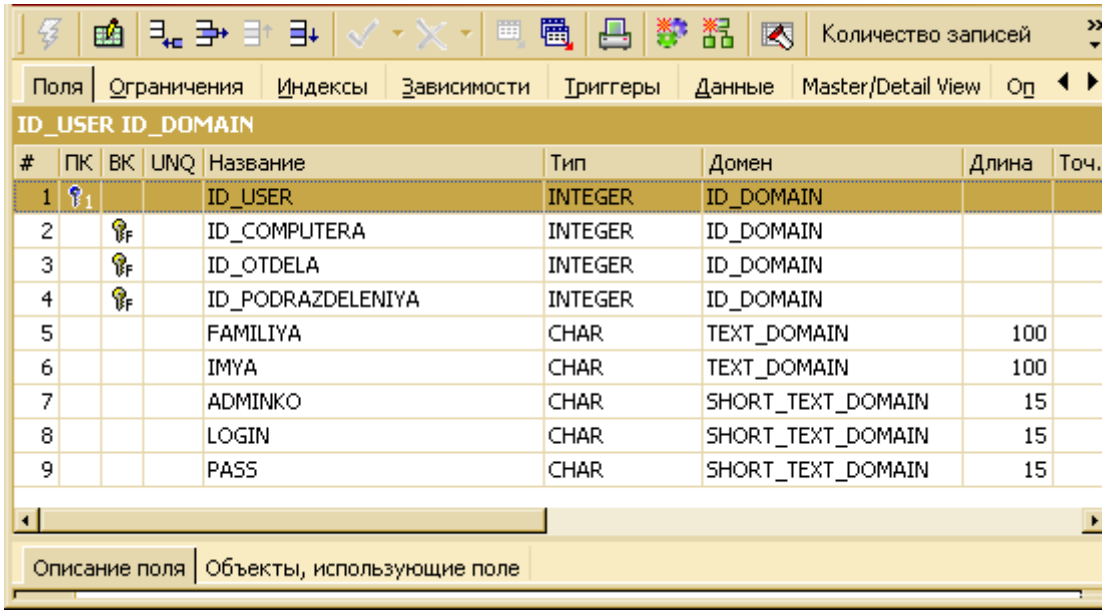

Рисунок 13 – Создание таблицы «USERS»

Когда все таблицы созданы, необходимо создать генераторы и триггеры для автоматического заполнения полей первичных ключей. Далее на рисунках 14 и 15 приведен пример создания генератора и его использования в триггере.

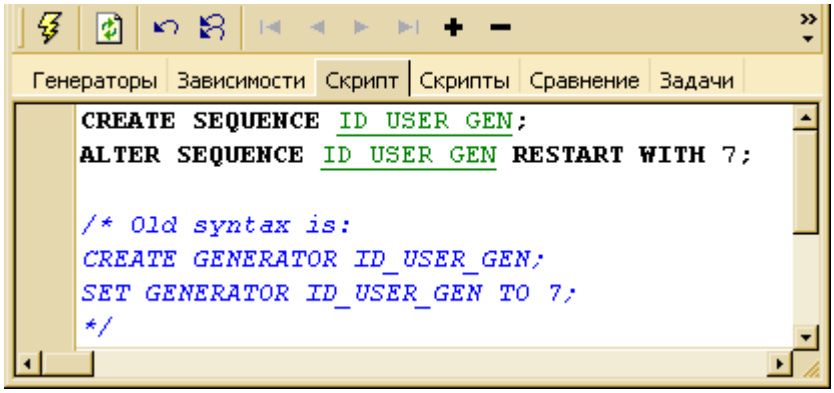

Рисунок 14 – Создание генератора «ID_USER_GEN»

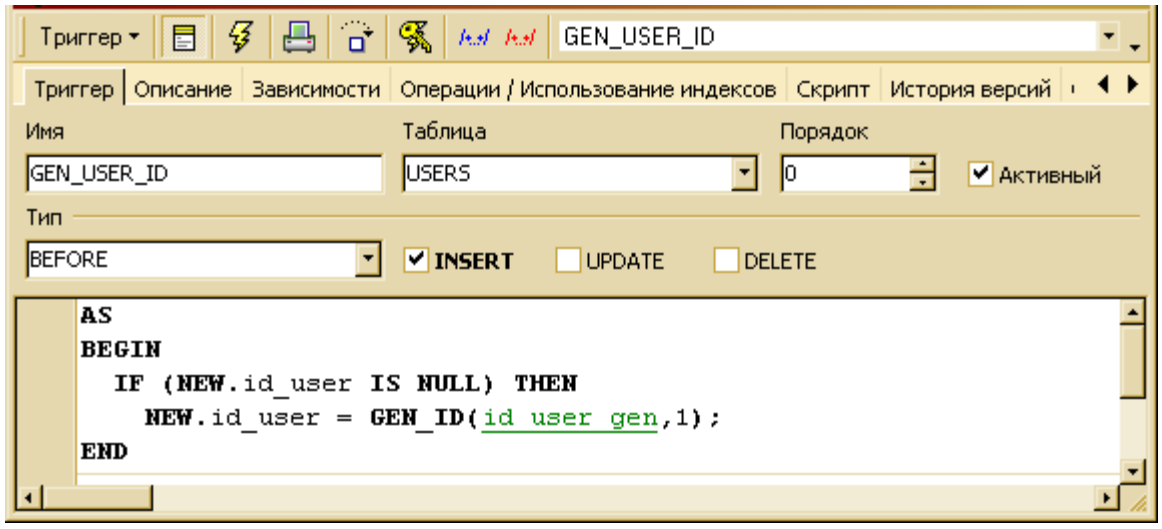

Рисунок 15 – Создание триггера «GEN_USER_ID»

Затем создадим необходимые для визуализации данных представления. На рисунке 16 приведен пример создания представления «ALL_USERS_VIEW». Данное представление предназначено для отображения полной информации о всех зарегистрированных в БД пользователях с заменой полей типа «ID» значимыми полями соответствующих таблиц.

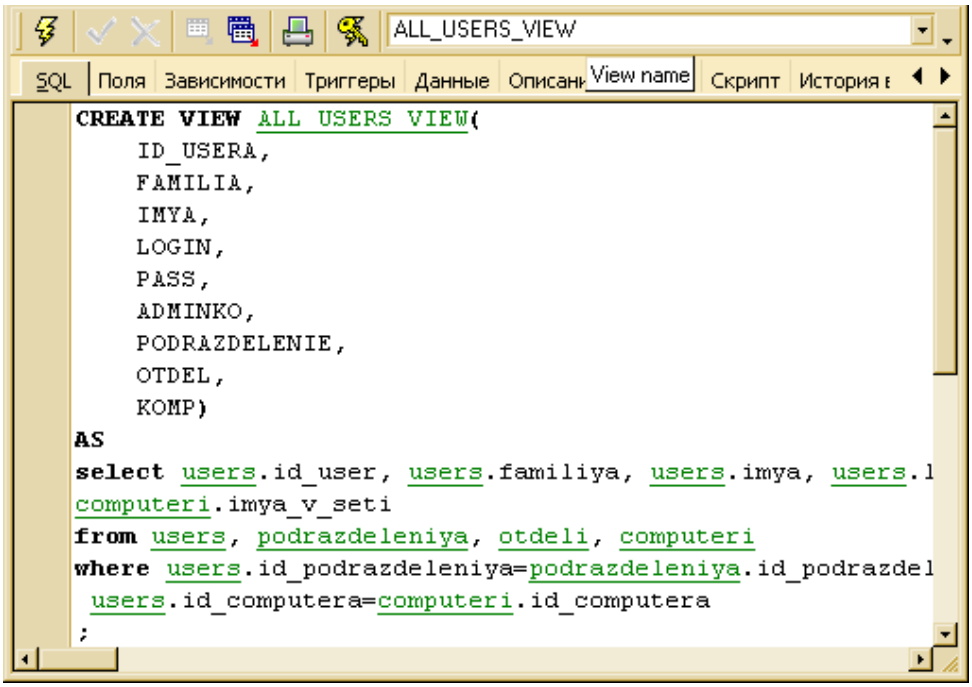

Рисунок 16 – Создание представления «ALL_USERS_VIEW»

Так как в случае данного проекта целесообразно было разгрузить клиенты от обработки данных, вся бизнес-логика была реализована с помощью сохраненных процедур. Пример создания одной из процедур приведен на рисунке 17.

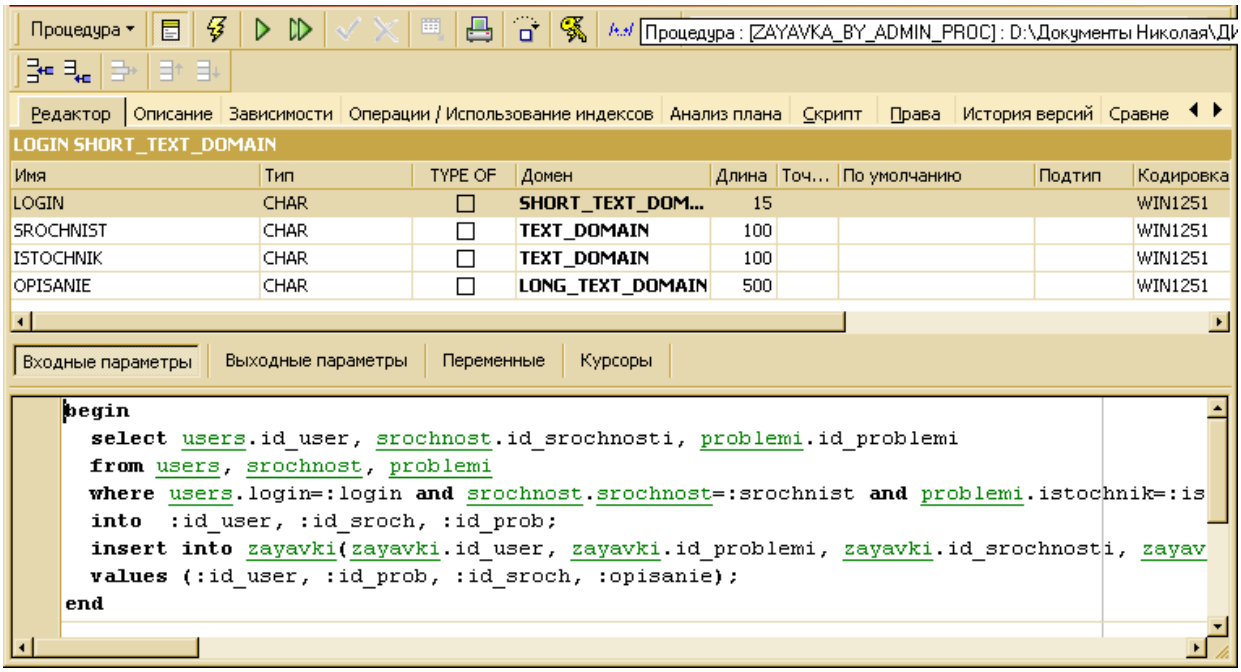

Рисунок 17 – Создание процедуры «ZAYAVKA_BY_ADMIN_PROC»

Перед написанием тела процедуры важно создать все требующиеся входные и выходные параметры, а также локальные переменные. Использование сохраненных процедур будет рассмотрено в разделе «Разработка пользовательского интерфейса».

Итоговый набор объектов базы данных представлен на рисунке 18.

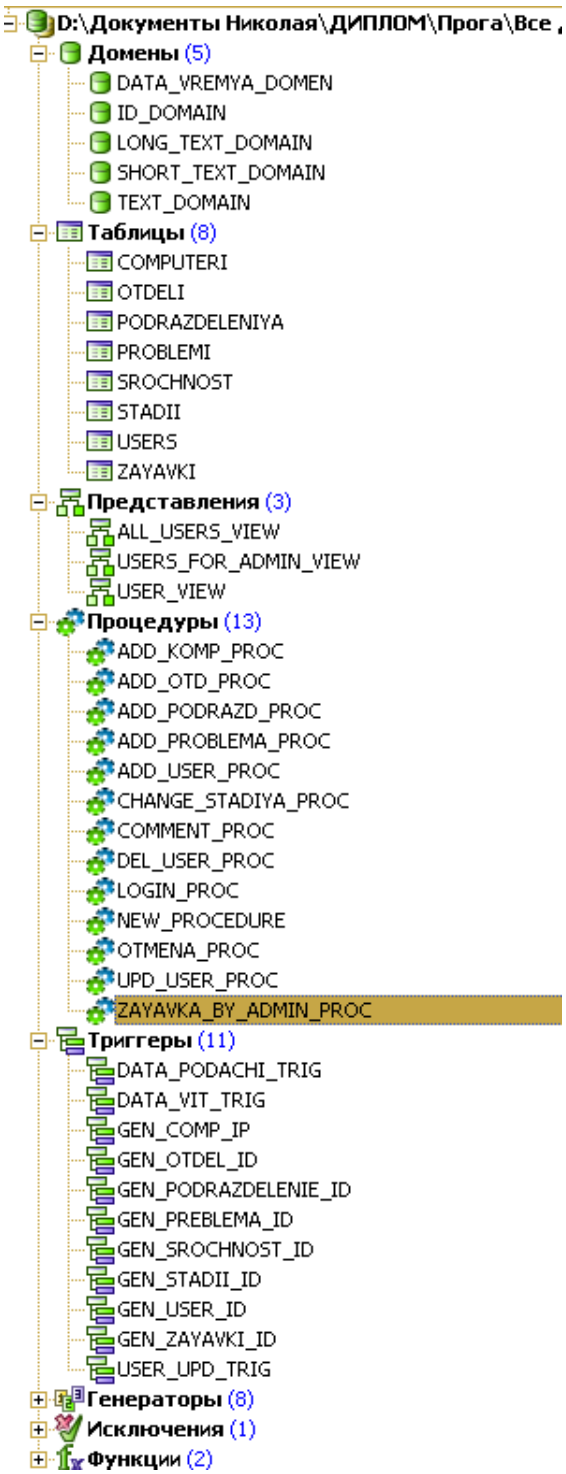

Рисунок 18 – Дерево объектов базы данных

3.3 Разработка пользовательского интерфейса

Первоначально необходимо создать модуль данных, расположить на нем и настроить все компоненты доступа к базе данных. Первым объектом является IBDataBase. Затем на него настраиваем IBTransaction. Для доступа к данным в базе размещаем и настраиваем компоненты IBTable и IBQuery. На них в свою очередь настраиваем соответствующие объекты представления данных DataSource. Чтобы завершить построение модуля данных, размещаем требуемое количество объектов IBStoredProc для реализации функция обработки данных через сохраненные в БД процедуры. Результат представлен на рисунке 19.

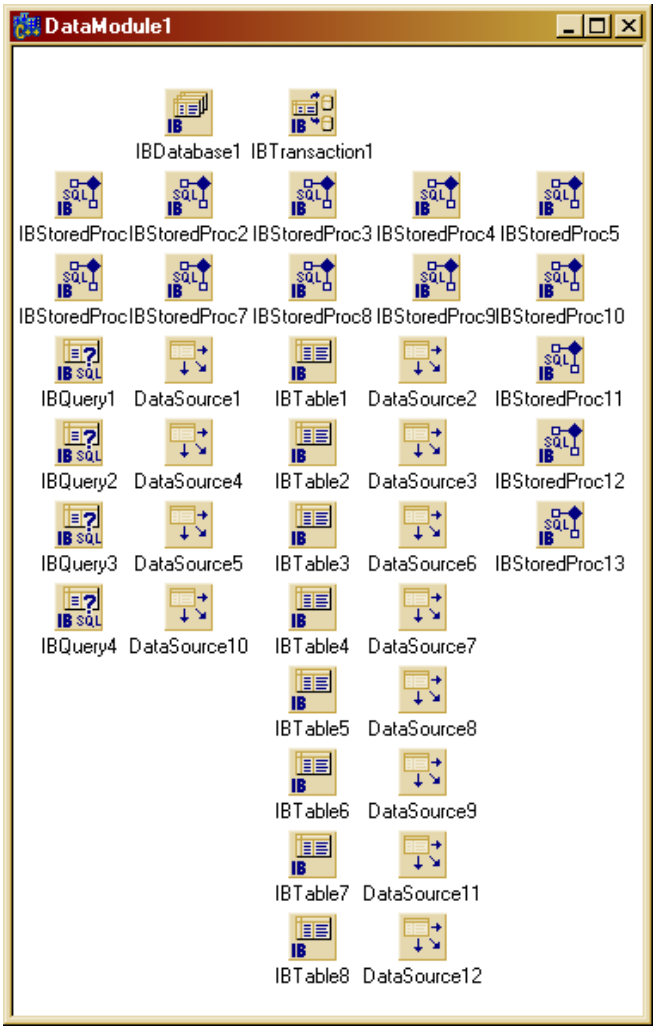

Рисунок 19 – DataModule1

Интерфейс программного средства «Подсистема учета заявок на обслуживание компьютерной техники корпорации «ЖБК-1» включает три экранные формы. Первая форма (рисунок 20) предназначена для подключения к базе данных и авторизации в ней пользователя.

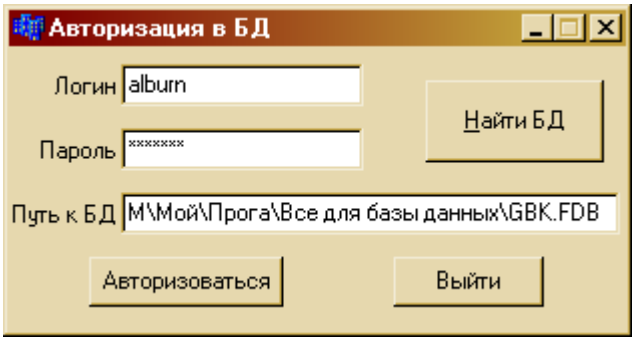

Рисунок 20 – Форма авторизации в БД

Кнопка «Найти БД» позволяет указать путь к файлу базы данных. При нажатии кнопки «Авторизоваться» происходит подключение к базе данные и выполнение сохраненной процедуры, которая определяет, обладает ли пользователь правами администратора. Если пользователь не обладает правами администратора, то открывается форма пользователя (рисунок 21).

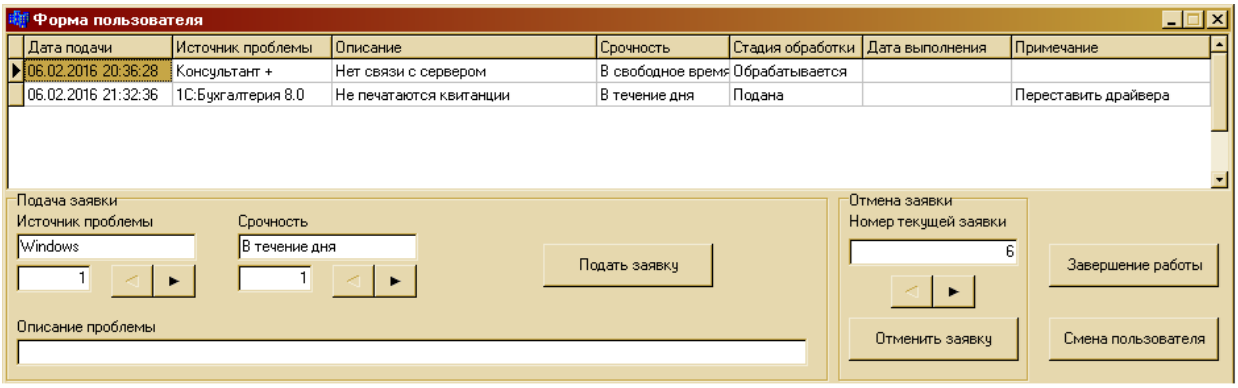

Рисунок 21 – Форма пользователя

На данной форме отражается результат параметрического запроса. В качестве параметра передается регистратор пользователя при авторизации. Таким образом, пользователь может просмотреть, подать или отменить только свои заявки и не имеет доступа к заявкам других пользователей.

Данная форма спроектирована максимально просто с целью минимизации требований к квалификации пользователя.

Если же пользователь обладает правами администратора, то видимой становится форма администратора, представленная на рисунке 22.

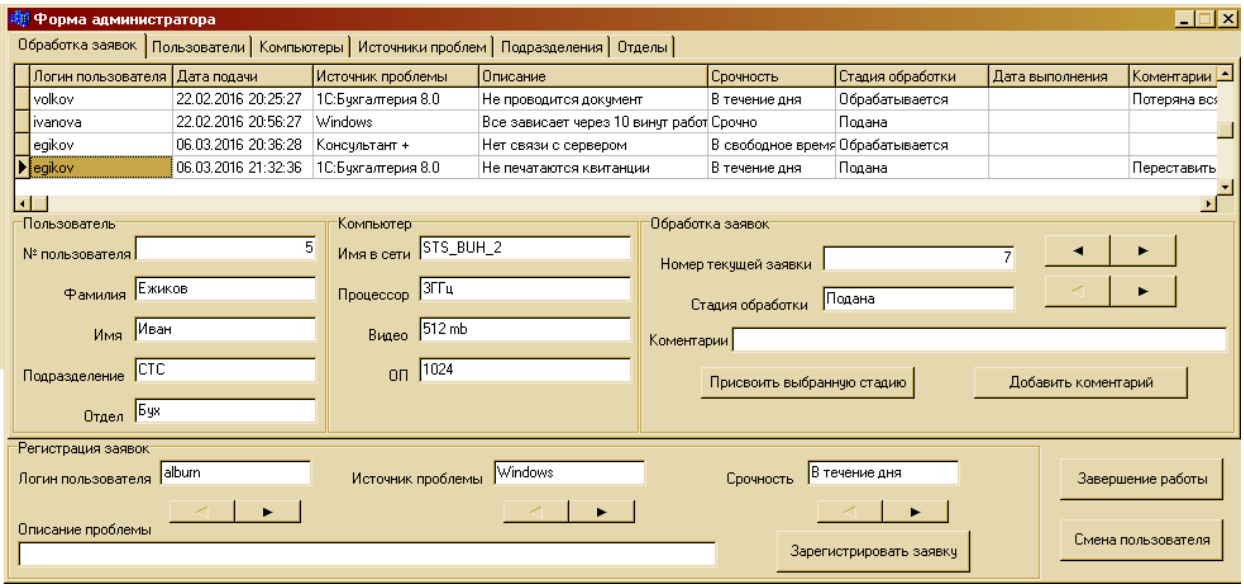

Рисунок 22 – Форма администратора

Данная форма представляет собой набор закладок в верхней части и поле для регистрации заявок в нижней. Последние пять закладок имеют сходную структуру и предназначены для работы с соответствующими таблицами БД. Наибольший интерес представляет закладка «Обработка заявок». В табличной части выводится список всех зарегистрированных когда-либо заявок. Группы компонентов DBEdit «Пользователь» и «Компьютер» предоставляют администратору расширенную информацию о выбранной заявке. Данная информация позволяет администратору определить местонахождение проблемного компьютера, а также дает базовое представление о его комплектации. Данная закладка позволяет администратору менять стадии обработки заявок и оставлять комментарии.

3.4 Описание контрольного примера реализации проекта

При создании любого программного обеспечения одним из основных этапов является этап тестирования. На данном этапе согласно сформулированным требованиям к системе проводится анализ ее работоспособности, надежности, устойчивости к сбоям. На основании результатов машинного эксперимента систему либо признают готовой к внедрению и дальнейшему использованию, либо отправляют на доработку.

База данных будет расположена на сервере фирмы. Клиентские компьютеры будут удаленно подключаться к ней.

Для начала попробуем запустить приложение, пройти авторизацию в БД и подать заявку от пользователя. Данные процессы иллюстрируются рисунками 23 и 24.

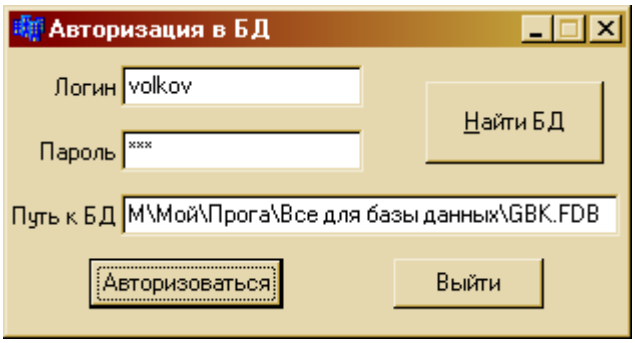

Рисунок 23 – Авторизация пользователя VOLKOV

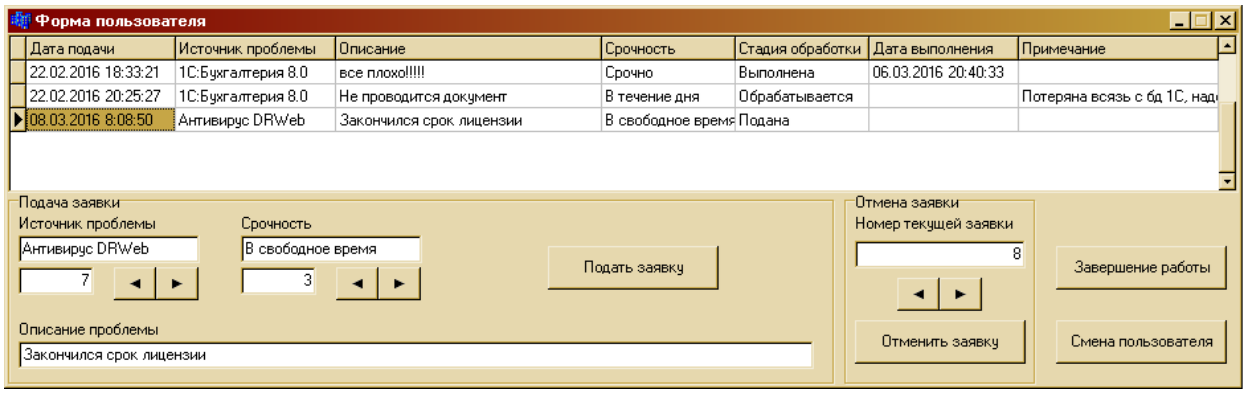

Рисунок 24 – Подача заявки на продление лицензии DRWeb

Теперь воспользуемся кнопкой «Смена пользователя» и авторизуемся с правами администратора. Отсортируем таблицу по дате подачи и найдем поданную заявку (рисунок 25).

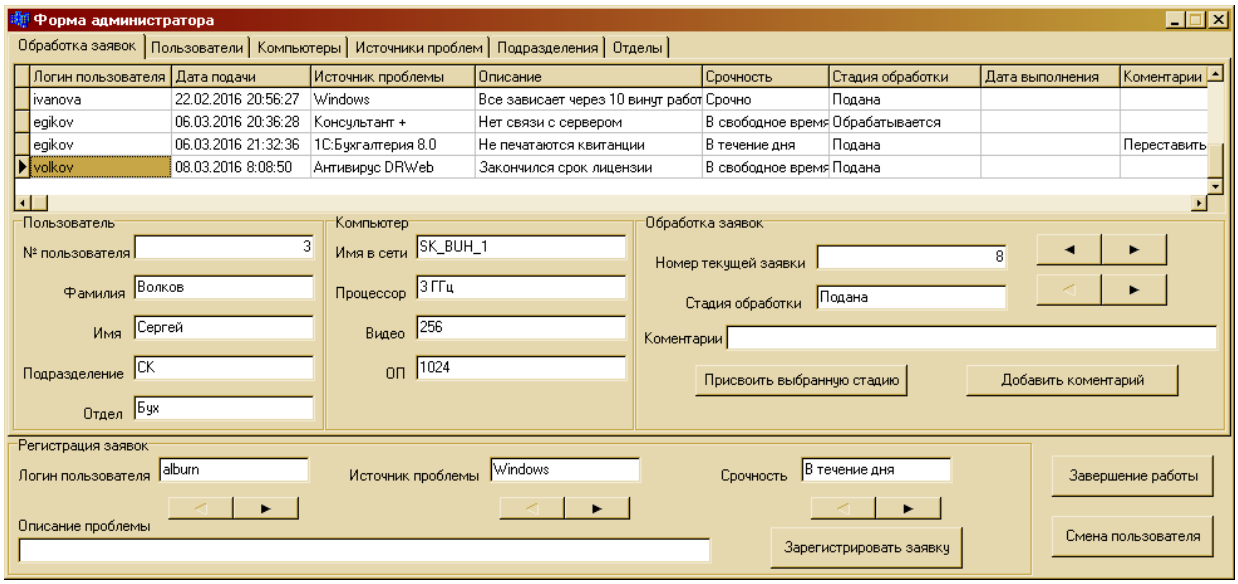

Рисунок 25 – Просмотр заявок администратором

В данном окне мы видим что Волков Сергей, бухгалтер из СтройКолора, подал заявку на продление лицензии. Отложим данную заявку объяснив это отсутствием новой лицензии. Результат представлен на рисунке 26.

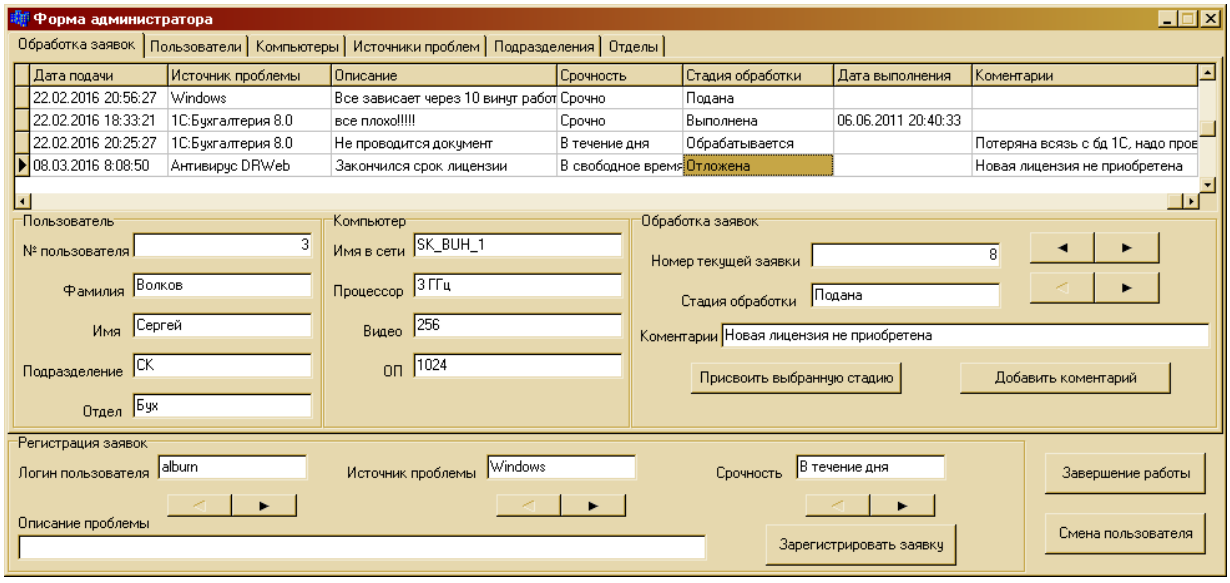

Рисунок 26 – Заявка отложена

4 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКАЯ ЧАСТЬ

4.1 Целесообразность разработки с экономической точки зрения

Главной задачей внедрения любой автоматизированной системы является увеличение прибыли внедряющего предприятия. Поэтому вопрос экономической целесообразности разработки является одним из важнейших.

Любой разрабатываемый программный продукт с точки зрения разработчика должен либо приносить доход от реализации его на рынке (если проект коммерческий), либо снижать издержки внедрившего данный продукт предприятия (если проект не предназначен для реализации на рынке). Подсистема учета заявок на обслуживание компьютерной техники разработана специально для отдела информационных технологий ЗАО «СофтКоннект». Поэтому реализация данного программного средства на рынке не предусмотрена. Однако его внедрение позволит извлечь выгоду для компании в целом.

Внедрение программного продукта «Подсистема учета заявок на обслуживание компьютерной техники позволит:

- повысить оперативность выполняемых работ;
- повысить производительность труда обслуживающего персонала;
- повысить производительность труда конечных пользователей;
- снизить затраты на деятельность отдела информационных технологий;
- проводить на основе данных БД аналитическую деятельность.

С экономической точки зрения достоинства разработки заключаются в оптимизации деятельности отдела информационных технологий ЗАО «СофтКоннект», что приведет не только к снижению издержек, но и

повышению качества обслуживания в целом, что окажет положительное влияние на производительность труда конечных пользователей.

4.2 SWOT – анализ разработки автоматизированной подсистемы учета заявок на обслуживание компьютерной техники

В настоящее время SWOT-анализ применяется достаточно широко в различных сферах экономики и управления. Его универсальность позволяет использовать его на различных уровнях и для различных объектов: анализ продукции, предприятия, конкурентов, города, региона и т.д. Этот метод как инструмент управленческого обследования (управленческого анализа) можно использовать для любого предприятия, чтобы предотвратить его попадание в кризисную ситуацию. Технология SWOT-анализа, как ее чаще всего отражают в учебной и специальной литературе, заключается в характеристике:

- внутренней среды (с выделением сильных и слабых сторон) и
- внешней среды (с выделением возможностей и угроз) предприятия описание выполняется с помощью факторов, не имеющих количественной оценки.

Факторы сводятся в таблицу; по значимости, как правило, не ранжируются.

Как показывает практика многие аналитики и экономисты-менеджеры, работающие в режиме недопущения предприятия до кризисного состояния, используют именно этот вариант, недопонимая исследовательского предназначения SWOT-анализа. Другие специалисты прекрасно понимают возможности SWOT-анализа, но в условиях кризиса, когда ощущается резкий дефицит времени для принятия решения по его урегулированию и давление интересантов, вынуждены проводить анализ в поверхностном варианте. Как правило, и в первом и во втором случаях исследование просто обрывается на

середине, ограничившись лишь простым описанием внутренней и внешней среды предприятия, хотя именно на этом этапе самое интересное только и начинается. Только на первый взгляд SWOT-анализ может освоить любой начинающий аналитик (не требуется знаний математики, теории вероятности, статистики и т.д.). На самом деле здесь нужны системные экономические знания, опыт и интуиция. Заметим, что в научной литературе содержание SWOT-анализа предполагает в дополнение к двум вышеназванным моментам (внутренняя, внешняя среда) и составление матрицы SWOT-анализа. Имеются и другие модификации. Развитие теории анализа и антикризисного управления требует совершенствования SWOT-анализа.

Для реализованного проекта подсистемы учета заявок на обслуживание компьютерной техники swot-матрица показана в таблице 4.

Таблица № 4 - Матрица SWOT-анализа для рассматриваемого проекта

Сильные стороны	Возможности		Угрозы.	Итого
	1. Повышение качества обслуживание	2. Повышение производительности труда сотрудников		
1. Низкая стоимость разработки.	0	0		0
2. Простота реализации	0	+		+1
3. Высокая скорость работы.	++	++		+4
4. Низкая вероятность ошибки.	++	++		+4
Итого	+4	+5		+9
Слабые стороны				
1. Узкоспециализированность	0	0		0
2. Необходимость сопровождения системы	0	– –		–2
3. Дополнительная нагрузка на аппаратные средства	–	–		–2
Итого	–1	–3		–4
Общий итог	+3	+2		+5

Проанализировав представленную таблицу, можно сделать вывод, что проект автоматизированной подсистемы учета заявок на обслуживание компьютерной техники является достаточно перспективным и имеет право на существование. Из таблицы 4 видно, что хотя проект и не направлен на снижение угроз, он достаточно неплохо способствует реализации дополнительных возможностей. Наиболее важными сильными сторонами являются высокая скорость работы и низкая вероятность возникновения ошибок. Существенное влияние на реализацию возможностей оказывают такие слабые стороны как дополнительная нагрузка на аппаратные средства и

необходимость сопровождения. Однако их влияние в данном случае можно признать приемлемым, так как в итоге возможности реализуются в достаточном объеме.

4.3 Калькуляция себестоимости научно-технической продукции

Любое предприятие в процессе своей деятельности совершает затраты на производство и реализацию продукции, на простое и расширенное воспроизводство основных фондов и оборотных средств, на социальное развитие трудового коллектива, на административные нужды и др.

Проект предполагает использование разработанной информационной системы, что подразумевает использование ЭВМ. Однако данная подсистема разрабатывается на сотрудников, чьи рабочие места уже оснащены необходимым аппаратным обеспечением. Также на серверах и рабочих станциях ООО «СщфтКоннект» уже используется все необходимое для внедрения подсистемы программное обеспечение. Поэтому основной статьей затрат при разработке подсистемы учета заявок на обслуживание компьютерной техники будет являться заработная плата разработчика. Распишем все затраты на проект разработки подсистемы в соответствии с утв. Миннауки от 15.06.1994 РФ №ОР-22-2-46 – форма 1-пн.

При разработке подсистемы использовались следующие затраты на оплату труда - месячный заработок – 16000 рублей в течение 3 месяцев (в расчет берется 30 дней в месяце). Всего за период реализации проекта разработчиком было получено 48000 рублей. В том числе:

- отчисления в Пенсионный фонд РФ составили:

16000 * 28% = 4480 рублей (отчисления в месяц).

4480 * 3= 13440 рублей (отчисления за весь период).

- отчисления в Фонд социального страхования РФ составили:

48000 * 4% = 1920 рублей.

- отчисления в Федеральный Фонд обязательного медицинского страхования составили:

48000 * 0.2% = 96 рублей.

- отчисления в Территориальный Фонд обязательного медицинского страхования составили:

48000 * 3.4% = 1632 рубля.

Всего отчисления на социальные нужды составят – 17088 рублей.

Таким образом составим таблицу калькуляции затрат на разработку подсистемы учета и обработки заявок на покупку оборудования и ТМЦ:

Таблица 5 - Калькуляция затрат на разработку подсистемы учета заявок на обслуживание компьютерной техники

	Наименование статей затрат	Сумма (руб.)
	Материалы и оборудование	0
	Спецоборудование для научных (экспериментальных) работ	0
	Затраты на оплату труда работников, непосредственно занятых созданием научно-технической продукции	48000
	Отчисления на социальные нужды	17088
	Прочие прямые расходы	0
	Накладные расходы	0
	Итого:	65088
	Затраты по работам, выполняемым сторонними организациями и предприятиями	0
	Всего себестоимость	65088

Если говорить о доходности проекта, то в данном случае можно выделить следующие аспекты:

- снижение расходов на бумажные носители;

- снижение расходов на связь;

- повышение производительности труда сотрудников ИТ отдела;

- повышение производительности труда конечных пользователей.

Выгода от роста производительности труда в масштабах компании можно предположить что проект окупиться в течении срока от нескольких месяцев до года. Таким образом, можно уверенно сказать, что проект является экономически выгодным и целесообразным.

ЗАКЛЮЧЕНИЕ

В заключение отметим, что в настоящее время автоматизация деятельности организаций позволяет добиться значительных конкурентных преимуществ вследствие как уменьшения затрат на ведение деятельности предприятия, так и повышения общей эффективности бизнес-процессов данного предприятия.

В процессе выполнения ВКР был проведен анализ уже существующих автоматизированных подсистем, внимательно рассмотрены и изучены их функциональные возможности. Проанализированы пробелы и недостатки подсистем. В результате была обоснована необходимость разработки дополнительной подсистемы, которая должна закрыть пробелы в системе автоматизации предприятия, сформулированы требования к ее функциональности.

При проведении первоначального анализа было осуществлено моделирование текущего способа реализации бизнес-процессов, определены недостатки и пути их устранения. На основании чего был проведен реинжиниринг, результатами которого стали модели реализации бизнес-процессов с использованием средств автоматизированной обработки данных, а также проект автоматизированной подсистемы.

На основании всех проделанных проектных работ, с учетом поставленных требований и указаний запланированной модели будущего проекта, была разработана автоматизированная подсистема «Учета заявок на обслуживание компьютерной техники». После чего система протестирована по различным критериям.

Подсистема учета заявок в процессе тестирования продемонстрировала свою работоспособность и отказоустойчивость при всех условиях эксплуатации, отвечающих базовым требованиям. Пользовательский модуль доказал эргономичность своего интерфейса и низкие требования к

квалификации конечных пользователей. Модуль администратора способствовал оптимизации учета пользователей, и их заявок.

Внедрение подсистемы «Учета заявок на обслуживание компьютерной техники корпорации» поможет значительно оптимизировать деятельность отдела информационных технологий. Внедрение базы данных позволит не только централизовано и в удобной форме хранить данные о заявках, но также проводить аналитическую деятельность с целью выявления недостатков в подготовке пользователей АРМ ЗАО «СофтКоннект», в материально-техническом их обеспечении. В итоге разработанная подсистема позволит повысить качество обслуживания конечных пользователей, сократить временные затраты, а также поток бумажных документов.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Автоматизированные информационные технологии в экономике: Учебник/ Под ред. Проф. Г.А. Титоренко. - М.: Компьютер, ЮНИЩ 1998.
2. Вендров А.М. CASE - технологии. Современные методы и средства проектирования информационных систем. - М.: Финансы и статистика, 1998.
3. Маклаков С.В. BFWin и ERWin. CASE-средства разработки информационных систем. М.: ДИАЛОГ-МИФИ, 2000.
4. Borland C++ Builder в подлиннике. А. Хомоненко. СПб: BHV, 2003 – 1216 стр.
5. Borland C++ Builder. Советы программистов (2-е издание): В.Озеров. – СПб: Символ-Плюс, 2002. – 976 стр.
6. Borland Delphi 6. Руководство разработчика: С.Тейксейра, К.Пачеко. – М: Вильямс, 2002. – 1120 стр.
7. Принципы проектирования и разработки программного обеспечения. Учебный курс MCSD: Скотт Ф. Уилсон, Брюс Мэйплс, Тим Лэндгрейв. – М: Русская редакция, 2002. – 736стр.
8. Проектирование экономических информационных систем: Учебник/Г.Н.Смирнова, А.А.Сорокин, Ю.Ф.Тельнов. – М: Финансы и статистика, 2003. – 512стр.
9. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. М.: «Финансы и статистика»,2002.
10. Самоучитель UML. Эффективный инструмент моделирования информационных систем: А. Леоненков. – СПб: BHV, 2001. – 304стр.
11. Borland C++ Builder 6 на примерах/Под ред. Ю. С. Ковтанюка — К.: Издательство Юниор, 2003. — 384 с., ил.

12. Нестандартные приемы программирования на Borland C++ Builder. — СПб.: БХВ-Петербург, 2005. — 560 с : ил.
13. Объектно-ориентированный подход и диаграммы классов в UML [электронный ресурс] – Режим доступа: http://www.iti.bpbu.ru/publicationb.'i-u/Real UML.htm:
14. Основы проектирования реляционных баз данных [электронный ресурс] – Режим доступа:http://books.kulichki.net/data/sql/sqll/
15. Понимание SQL (Understanding SQL). [электронный ресурс] – Режим доступа: http://www.sql.ni/docs/sql/u sqI/index.shtml
16. Borland AML Portal. [электронный ресурс] – Режим доступа: http://www.almportal.ru
17. Компания Borland. [электронный ресурс] – Режим доступа: http://www.borland.com
18. Русскоязычный сайт компании Borland. [электронный ресурс] – Режим доступа: http://www.borland.ru

ПРИЛОЖЕНИЕ А

Листинг формы авторизации

```
#include <vcl.h>
#pragma hdrstop
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm1 *Form1;
//---------------------------------------------------------------------------
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm1::FileOpen1Accept(TObject *Sender)
{
Form1->LabeledEdit3->Text=Form1->FileOpen1->Dialog->FileName;
}
//---------------------------------------------------------------------------
void __fastcall TForm1::Button3Click(TObject *Sender)
{
Close();
}
//---------------------------------------------------------------------------
```

```
void __fastcall TForm1::Button2Click(TObject *Sender)
{
DataModule1->IBDatabase1->DatabaseName=Form1->LabeledEdit3->Text;
DataModule1->IBDatabase1->Connected=True;
DataModule1->IBTransaction1->Active=True;
DataModule1->IBStoredProc1->ParamByName("LOGIN")-
>AsString=Form1->LabeledEdit1->Text;
DataModule1->IBStoredProc1->ParamByName("PASS")->AsString=Form1-
>LabeledEdit2->Text;
DataModule1->IBStoredProc1->Prepare();
DataModule1->IBStoredProc1->ExecProc();
DataModule1->IBTransaction1->Commit();
if (DataModule1->IBStoredProc1->ParamByName("ADMINKO")-
>AsString=="нет ")
{DataModule1->IBQuery1->ParamByName("user_id")-
>Value=DataModule1->IBStoredProc1->ParamByName("ID_USER")->AsString;
DataModule1->IBStoredProc3->ParamByName("ID_USER")-
>Value=DataModule1->IBStoredProc1->ParamByName("ID_USER")->AsString;
Form1->Visible=False;
Form2->Visible=True;
DataModule1->IBQuery1->Active=False;
DataModule1->IBQuery1->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;}
else {
DataModule1->IBQuery2->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
```

```
        DataModule1->IBTable5->Active=True;
        DataModule1->IBTable6->Active=True;
        DataModule1->IBTable7->Active=True;
        DataModule1->IBTable8->Active=True;
        DataModule1->IBQuery3->ParamByName("id_user")->AsString=Form3-
>DBEdit6->Text;
        Form1->Visible=False;
        Form3->Visible=True;}
        DataModule1->IBQuery3->Active=False;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
        }
        //---------------------------------------------------------------------------
```

ПРИЛОЖЕНИЕ Б

Листинг формы пользователя

```
#include <vcl.h>
#pragma hdrstop
#include "Unit3.h"
#include "Unit1.h"
#include "Unit2.h"
#include "Unit4.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm2 *Form2;
//---------------------------------------------------------------------------
__fastcall TForm2::TForm2(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm2::FormClose(TObject *Sender, TCloseAction
&Action)
{
Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button2Click(TObject *Sender)
{
Form2->Visible=False;
Form1->Visible=True;
```

```
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button1Click(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button3Click(TObject *Sender)
{
DataModule1->IBStoredProc2->ParamByName("ID_ZAYAVKI")-
>AsString=Form2->DBEdit1->Text;
DataModule1->IBStoredProc2->Prepare();
DataModule1->IBStoredProc2->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery1->Active=False;
DataModule1->IBQuery1->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::Button4Click(TObject *Sender)
{
DataModule1->IBStoredProc3->ParamByName("ID_PROBLEMI")-
>AsString=Form2->DBEdit4->Text;
DataModule1->IBStoredProc3->ParamByName("ID_SROCHNOSTI")-
>AsString=Form2->DBEdit5->Text;
DataModule1->IBStoredProc3->ParamByName("OPISANIE")-
>AsString=Form2->Edit1->Text;
DataModule1->IBStoredProc3->Prepare();
DataModule1->IBStoredProc3->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery1->Active=False;
```

```
DataModule1->IBQuery1->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm2::DBGrid1TitleClick(TColumn *Column)
{
DataModule1->IBQuery1->Active=false;
DataModule1->IBQuery1->SQL->Clear();
DataModule1->IBQuery1->SQL->Add("SELECT * FROM USER_VIEW");
DataModule1->IBQuery1->SQL->Add("WHERE  ID_USER=:user_id");
DataModule1->IBQuery1->SQL->Add("ORDER BY");
DataModule1->IBQuery1->SQL->Add(Column->FieldName);
DataModule1->IBQuery1->ParamByName("user_id")-
>Value=DataModule1->IBStoredProc1->ParamByName("ID_USER")->AsString;
DataModule1->IBQuery1->Active=true;
}
//---------------------------------------------------------------------------
```

ПРИЛОЖЕНИЕ В

Листинг формы администратора

```
#include <vcl.h>
#pragma hdrstop
#include "Unit4.h"
#include "Unit1.h"
#include "Unit2.h"
#include "Unit3.h"
//---------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
TForm3 *Form3;
//---------------------------------------------------------------------------
__fastcall TForm3::TForm3(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------------
void __fastcall TForm3::FormClose(TObject *Sender, TCloseAction
&Action)
{
Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::DBGrid1CellClick(TColumn *Column)
{
DataModule1->IBQuery3->Active=False;
```

```
DataModule1->IBQuery3->ParamByName("id_user")->AsString=Form3->DBEdit6->Text;
DataModule1->IBQuery3->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button2Click(TObject *Sender)
{
Form3->Visible=False;
Form1->Visible=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button1Click(TObject *Sender)
{
Form1->Close();
}
//---------------------------------------------------------------------------
void __fastcall TForm3::DBNavigator1Click(TObject *Sender,
    TNavigateBtn Button)
{
DataModule1->IBQuery3->Active=False;
DataModule1->IBQuery3->ParamByName("id_user")->AsString=Form3->DBEdit6->Text;
DataModule1->IBQuery3->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button3Click(TObject *Sender)
{
DataModule1->IBStoredProc4->ParamByName("ID_ZAYAVKI")->AsString=Form3->DBEdit11->Text;
```

```
        DataModule1->IBStoredProc4->ParamByName("STADIYA")-
>AsString=Form3->DBEdit12->Text;
        DataModule1->IBStoredProc4->Prepare();
        DataModule1->IBStoredProc4->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
        DataModule1->IBTable1->Active=True;
        DataModule1->IBTable2->Active=True;
        DataModule1->IBTable3->Active=True;
        DataModule1->IBTable4->Active=True;
        DataModule1->IBTable5->Active=True;
        DataModule1->IBTable6->Active=True;
        DataModule1->IBTable7->Active=True;
        DataModule1->IBTable8->Active=True;
        }
        //---------------------------------------------------------------------
        void __fastcall TForm3::Button4Click(TObject *Sender)
        {
        DataModule1->IBStoredProc5->ParamByName("ID_ZAYAVKI")-
>AsString=Form3->DBEdit11->Text;
        DataModule1->IBStoredProc5->ParamByName("COMMENT")-
>AsString=Form3->LabeledEdit1->Text;
        DataModule1->IBStoredProc5->Prepare();
        DataModule1->IBStoredProc5->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
```

```
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button5Click(TObject *Sender)
{
DataModule1->IBStoredProc6->ParamByName("LOGIN")-
>AsString=Form3->DBEdit13->Text;
DataModule1->IBStoredProc6->ParamByName("SROCHNIST")-
>AsString=Form3->DBEdit15->Text;
DataModule1->IBStoredProc6->ParamByName("ISTOCHNIK")-
>AsString=Form3->DBEdit14->Text;
DataModule1->IBStoredProc6->ParamByName("OPISANIE")-
>AsString=Form3->LabeledEdit2->Text;
DataModule1->IBStoredProc6->Prepare();
DataModule1->IBStoredProc6->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery2->Active=True;
DataModule1->IBQuery3->Active=True;
DataModule1->IBQuery4->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
```

```
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;

}
//---------------------------------------------------------------------------
void __fastcall TForm3::DBGrid1TitleClick(TColumn *Column)
{
DataModule1->IBQuery2->Active=false;
DataModule1->IBQuery2->SQL->Clear();
DataModule1->IBQuery2->SQL->Add("select * from USER_VIEW");
DataModule1->IBQuery2->SQL->Add("ORDER BY");
DataModule1->IBQuery2->SQL->Add(Column->FieldName);
DataModule1->IBQuery2->Active=true;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button6Click(TObject *Sender)
{
DataModule1->IBStoredProc7->ParamByName("PROBLEMA")-
>AsString=Form3->LabeledEdit3->Text;
DataModule1->IBStoredProc7->Prepare();
DataModule1->IBStoredProc7->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery2->Active=True;
DataModule1->IBQuery3->Active=True;
DataModule1->IBQuery4->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
```

```
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button7Click(TObject *Sender)
{
DataModule1->IBStoredProc8->ParamByName("NAME")-
>AsString=Form3->LabeledEdit4->Text;
DataModule1->IBStoredProc8->ParamByName("OP")->AsString=Form3-
>LabeledEdit5->Text;
DataModule1->IBStoredProc8->ParamByName("PROC")-
>AsString=Form3->LabeledEdit6->Text;
DataModule1->IBStoredProc8->ParamByName("VIDEO")-
>AsString=Form3->LabeledEdit7->Text;
DataModule1->IBStoredProc8->Prepare();
DataModule1->IBStoredProc8->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery2->Active=True;
DataModule1->IBQuery3->Active=True;
DataModule1->IBQuery4->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
```

```
DataModule1->IBTable8->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button8Click(TObject *Sender)
{
DataModule1->IBStoredProc9->ParamByName("ID_USER")->AsString=Form3->DBEdit3->Text;
DataModule1->IBStoredProc9->Prepare();
DataModule1->IBStoredProc9->ExecProc();
DataModule1->IBTransaction1->Commit();
DataModule1->IBQuery2->Active=True;
DataModule1->IBQuery3->Active=True;
DataModule1->IBQuery4->Active=True;
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;
}
//---------------------------------------------------------------------------
void __fastcall TForm3::DBGrid4TitleClick(TColumn *Column)
{
DataModule1->IBQuery4->Active=false;
DataModule1->IBQuery4->SQL->Clear();
DataModule1->IBQuery4->SQL->Add("select * from ALL_USERS_VIEW");
DataModule1->IBQuery4->SQL->Add("ORDER BY");
```

```
        DataModule1->IBQuery4->SQL->Add(Column->FieldName);
        DataModule1->IBQuery4->Active=true;
        }
        //---------------------------------------------------------------------
        void __fastcall TForm3::Button10Click(TObject *Sender)
        {
        DataModule1->IBStoredProc10->ParamByName("FAMIL")-
>AsString=Form3->LabeledEdit8->Text;
        DataModule1->IBStoredProc10->ParamByName("IMYA")-
>AsString=Form3->LabeledEdit9->Text;
        DataModule1->IBStoredProc10->ParamByName("LOGIN")-
>AsString=Form3->LabeledEdit10->Text;
        DataModule1->IBStoredProc10->ParamByName("PASS")-
>AsString=Form3->LabeledEdit11->Text;
        DataModule1->IBStoredProc10->ParamByName("ADMINKO")-
>AsString=Form3->LabeledEdit12->Text;
        DataModule1->IBStoredProc10->ParamByName("PODRAZD")-
>AsString=Form3->DBEdit18->Text;
        DataModule1->IBStoredProc10->ParamByName("OTDEL")-
>AsString=Form3->DBEdit16->Text;
        DataModule1->IBStoredProc10->ParamByName("KOMP")-
>AsString=Form3->DBEdit17->Text;
        DataModule1->IBStoredProc10->Prepare();
        DataModule1->IBStoredProc10->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
        DataModule1->IBTable1->Active=True;
        DataModule1->IBTable2->Active=True;
```

```
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;
Form3->LabeledEdit8->Text="";
Form3->LabeledEdit9->Text="";
Form3->LabeledEdit10->Text="";
Form3->LabeledEdit11->Text="";
Form3->LabeledEdit12->Text="";
}
//---------------------------------------------------------------------------
void __fastcall TForm3::Button9Click(TObject *Sender)
{
DataModule1->IBStoredProc11->ParamByName("FAMIL")-
>AsString=Form3->LabeledEdit8->Text;
DataModule1->IBStoredProc11->ParamByName("IMYA")-
>AsString=Form3->LabeledEdit9->Text;
DataModule1->IBStoredProc11->ParamByName("LOGIN")-
>AsString=Form3->LabeledEdit10->Text;
DataModule1->IBStoredProc11->ParamByName("PASS")-
>AsString=Form3->LabeledEdit11->Text;
DataModule1->IBStoredProc11->ParamByName("ADMINKO")-
>AsString=Form3->LabeledEdit12->Text;
DataModule1->IBStoredProc11->ParamByName("PODRAZD")-
>AsString=Form3->DBEdit18->Text;
DataModule1->IBStoredProc11->ParamByName("OTDEL")-
>AsString=Form3->DBEdit16->Text;
```

```
        DataModule1->IBStoredProc11->ParamByName("KOMP")-
>AsString=Form3->DBEdit17->Text;
        DataModule1->IBStoredProc11->ParamByName("ID_USER")-
>AsString=Form3->DBEdit3->Text;
        DataModule1->IBStoredProc11->Prepare();
        DataModule1->IBStoredProc11->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
        DataModule1->IBTable1->Active=True;
        DataModule1->IBTable2->Active=True;
        DataModule1->IBTable3->Active=True;
        DataModule1->IBTable4->Active=True;
        DataModule1->IBTable5->Active=True;
        DataModule1->IBTable6->Active=True;
        DataModule1->IBTable7->Active=True;
        DataModule1->IBTable8->Active=True;
        Form3->LabeledEdit8->Text="";
        Form3->LabeledEdit9->Text="";
        Form3->LabeledEdit10->Text="";
        Form3->LabeledEdit11->Text="";
        Form3->LabeledEdit12->Text="";
        }
        //---------------------------------------------------------------------
        void __fastcall TForm3::Button11Click(TObject *Sender)
        {
        DataModule1->IBStoredProc12->ParamByName("NAIMEN")-
>AsString=Form3->LabeledEdit15->Text;
```

```
        DataModule1->IBStoredProc12->ParamByName("SHORT")-
>AsString=Form3->LabeledEdit16->Text;
        DataModule1->IBStoredProc12->Prepare();
        DataModule1->IBStoredProc12->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
        DataModule1->IBTable1->Active=True;
        DataModule1->IBTable2->Active=True;
        DataModule1->IBTable3->Active=True;
        DataModule1->IBTable4->Active=True;
        DataModule1->IBTable5->Active=True;
        DataModule1->IBTable6->Active=True;
        DataModule1->IBTable7->Active=True;
        DataModule1->IBTable8->Active=True;
        }
        //---------------------------------------------------------------------
        void __fastcall TForm3::Button12Click(TObject *Sender)
        {
        DataModule1->IBStoredProc13->ParamByName("NAIMEN")-
>AsString=Form3->LabeledEdit13->Text;
        DataModule1->IBStoredProc13->ParamByName("SHORT")-
>AsString=Form3->LabeledEdit14->Text;
        DataModule1->IBStoredProc13->Prepare();
        DataModule1->IBStoredProc13->ExecProc();
        DataModule1->IBTransaction1->Commit();
        DataModule1->IBQuery2->Active=True;
        DataModule1->IBQuery3->Active=True;
        DataModule1->IBQuery4->Active=True;
```

```
DataModule1->IBTable1->Active=True;
DataModule1->IBTable2->Active=True;
DataModule1->IBTable3->Active=True;
DataModule1->IBTable4->Active=True;
DataModule1->IBTable5->Active=True;
DataModule1->IBTable6->Active=True;
DataModule1->IBTable7->Active=True;
DataModule1->IBTable8->Active=True;
}
//---------------------------------------------------------------------------
```