ВВЕДЕНИЕ

Внедрение техники и технологии в область обработки информации привело к повышению производительности, сравнимому с тем, которое дали стандартизация и сборочные конвейеры в производстве в начале промышленной революции. Точно так же, как не выдержали конкуренции те организации, которые продолжали использовать старую технологию производства, в информационном обществе не смогут конкурировать организации, не использующие информационные технологии. Однако массовое использование в организациях компьютеров и другого оборудования порождает новые проблемы, которые быстро множатся по мере увеличения покупаемой оргтехники.

Особенно хорошо эти проблемы становятся заметны, когда речь идёт о крупном предприятии, имеющем в своём составе не одну сотню, а то и тысячу единиц техники и имеющем крупную географически распределённую структуру.

Серьёзной проблемой такой организации является учёт и систематизация используемого оборудования, его инвентаризация и обслуживание. Необходимо иметь большую базу данных используемой техники и удобный инструмент работы с ней.

Автоматизация учета на предприятии предполагает замену алгоритмизируемых операций, выполняемых человеком, на человеко-машинные, где человеку отводится роль оператора, использующего компьютерный инструментарий. Данные преобразования повышают эффективность работы как отдельных исполнителей, так и предприятия в целом. Это достигается путем использования особых механизмов организации и управления информацией, что повышает ее достоверность, полноту и оперативность представления.

Цель автоматизации в области управленческого учета – представление адекватной картины бизнеса и основы для принятия управленческих решений. На практике суть преобразований сводится к построению некой программной системы, обеспечивающей надежное хранение, быструю и точную обработку и удобное представление полной учетной информации.

Другой проблемой такой организации становится своевременный приём заявок от филиалов и отделений на ремонт и техническое обслуживание требующей этого техники, своевременная доставка такого оборудования в отдел технической поддержки предприятия, согласование сроков ремонтов и возвратов техники на рабочие места, постоянное наличие необходимых комплектующих, расходных материалов, автотранспорта.

Решением описанных проблем является автоматизация процесса приёма заявок на ремонт и техобслуживание вычислительной и оргтехники предприятия.

1 Описание объекта автоматизации

Внедрение предлагаемой АИС может быть осуществлено в любой организация, имеющая крупную сеть филиалов географически распределённых и имеющих единый центр технической поддержки. Такой центр обеспечивает поддержку работоспособности используемой в организации и соответственно во всех филиалах техники. Он отвечает за закупку оборудования, комплектующих и ЗИП, ремонт и диагностику вышедшего из строя оборудования, а также координирует действия ИТО в регионах. Условно такой центр можно назвать отделом технической поддержки (ОТП) предприятия.

1.1 Отдел технической поддержки ФГУП «Почта России»

В данном дипломном проекте в качестве примера предприятия, требующего внедрения АИС, рассматривается Российская федеральная почтовая сеть.

Российская федеральная почтовая сеть предоставляет услуги почтовой связи на всей территории Российской Федерации, включая все города и сельские населенные пункты.

«Почта России» сегодня – это 84 филиала, 42 000 объектов почтовой связи, 415 000 сотрудников. Ежегодно почтовые работники России принимают, обрабатывают и доставляют более 1,5 млрд писем, 48 млн посылок и более 190 млн. денежных переводов.

Почта России предлагает своим клиентам свыше 80 почтовых, финансовых, инфокоммуникационных и прочих услуг.

ФГУП «Почта России» было основано в рамках проведения реструктуризации организаций федеральной почтовой связи, направленной на обеспечение гарантированного предоставления универсальных услуг почтовой

связи на всей территории Российской Федерации, расширение спектра услуг почтовой связи, создание современной почтовой инфраструктуры.

Состав Ростовского отделения предприятия в среднем составляет десять тысяч человек. В его состав входит до двадцати филиалов и более тысячи отделений. Количество единиц техники, подлежащей учёту, составляет не менее пяти тысяч. До сих пор на предприятии не внедрено ни одной системы, позволяющей эффективно справляться с описанными выше трудностями.

Объектом автоматизации, является процесс учёта и исполнения заявок ремонта оргтехники. Под оргтехникой необходимо понимать следующие виды техники:

- вычислительная техника (системные блоки ПЭВМ, мониторы, серверы);
- оргтехника (принтеры, копиры, сканеры);
- контрольно-кассовые машины, терминалы;
- сетевое оборудование (коммутаторы, маршрутизаторы);
- банковское оборудование (счётчики и детекторы банкнот);
- средства связи (телефоны, модемы).

Ремонт включает в себя обслуживание всех неисправностей, возникающих в процессе её эксплуатации, технический осмотр, профилактические работы. Процесс исполнения заявок, помимо прочего, включает ведение баз данных, планирование различных мероприятий, связанных с обслуживанием техники.

1.2 Цели и задачи автоматизации отдела технической поддержки ФГУП «Почта России»

В ходе выполнения дипломной работы мной были проведены исследования среди персонала и руководства отдела технической поддержки (ОТП). Проанализировав собранный материал, можно представить текущую

структуру процесса функционирования подразделений ФГУП Почта России, в которые вовлечён ОТП (рисунок 1). Принимая решение о внедрении АИС, я буду придерживаться поставленных передо мной целей. Таким образом, цели автоматизации можно сформулировать следующим образом:

- повысить эффективность работы технических служб;
- сократить затраты на ремонт видов техники;
- сократить время простоя техники в ремонте;
- систематизировать информацию о имеющихся видах техники;
- ведение истории каждой единицы техники;
- контроль за расходом комплектующих и ЗИП;
- организовать электронный документооборот;
- организовать единую информационно справочную среду для технических подразделений.

Основной задачей является разработка АИС поддержания технических служб предприятия. Программная составляющая АИС должна быть кроссплатформенной и предоставлять доступ к единой системе из различных географических точек. Добиться поставленных целей можно путём автоматизации процесса учёта и исполнения заявок на ремонт оргтехники. При этом, информация поступающая в базу данных системы обеспечит достижение целей учёта и систематизации, а бизнес логика программного средства обеспечит достижение целей анализа, планирования, доступа к данным и других функций связанных с обработкой получаемой информации.

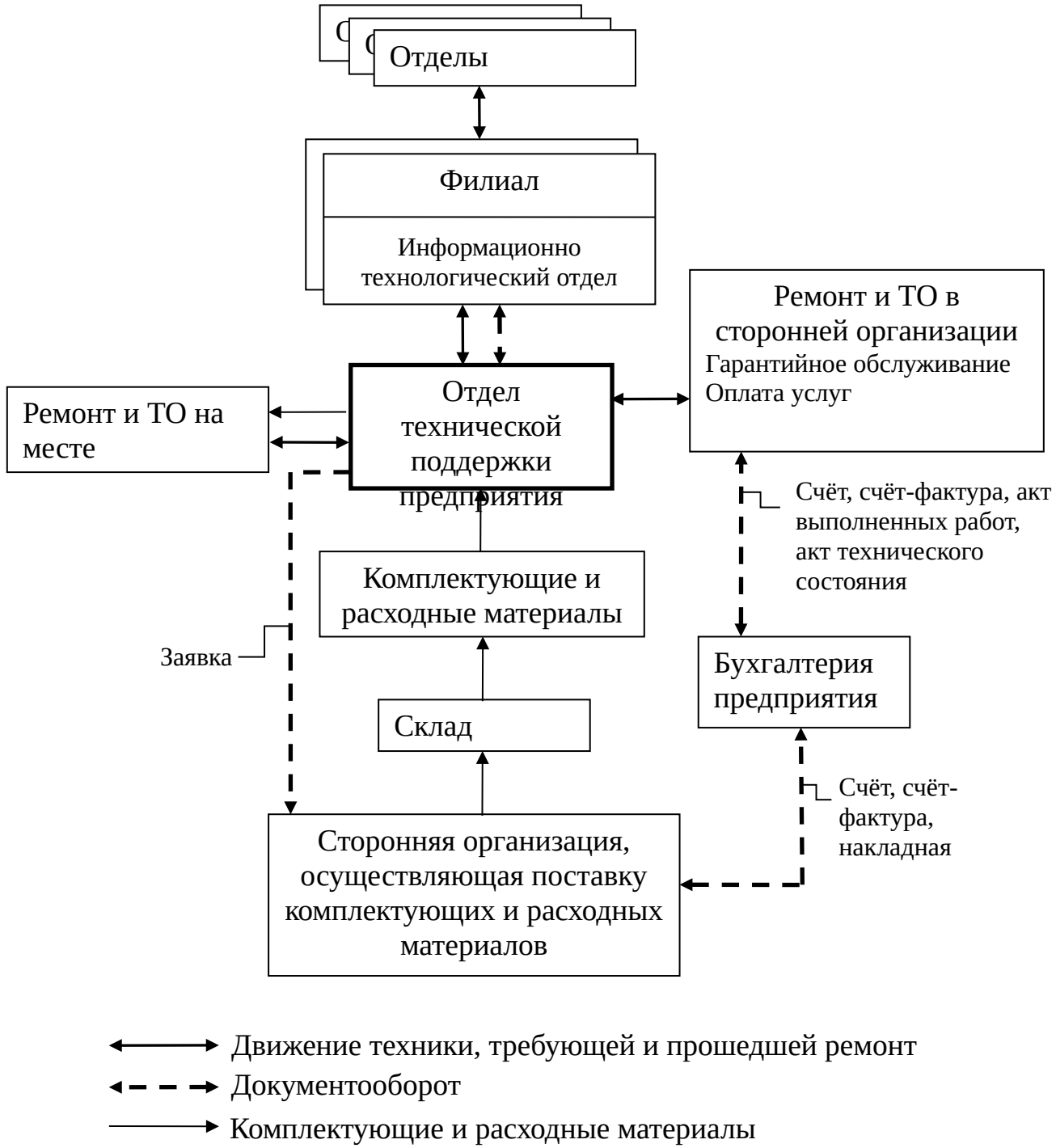

Рисунок 1. – Структурная схема автоматизируемого процесса

2 Выбор АИС

Рассмотрим, при помощи какой автоматизированной системы можно добиться поставленной цели.

Существует ряд готовых решений для автоматизации типовых задач определенных областей учета или же учета в целом. У полнофункциональных пакетов свои преимущества: их поэтапное внедрение позволяет не только автоматизировать большинство бизнес-процессов, но и естественным образом решать проблемы интеграции подсистем, унификации пользовательского интерфейса и т.д. [1].

Однако внедрение подобной системы является достаточно рискованным процессом по ряду причин, главные из которых – необходимость настройки программы на решение специфических задач конкретного предприятия и необходимость проведения организационных преобразований для адаптации самого учета к системе. Кроме того, если для автоматизации бухгалтерского учета подойдет готовое решение (например, система «1С:Предприятие»), требующее минимальной доработки, то в такой слабо формализованной области, как управленческий учет, возможно, более эффективным решением будет создание уникальной программной системы, максимально удовлетворяющей потребности конкретного предприятия. Чаще всего такое решение будет еще и менее громоздким. Однако при отсутствии удобного инструментария построение подобной системы может потребовать значительных затрат времени.

Конечно, у полнофункциональных пакетов есть свои преимущества: их поэтапное внедрение позволяет не только автоматизировать большинство бизнес-процессов, но и естественным образом решить проблемы интеграции подсистем, консолидации данных, унификации пользовательского интерфейса и т. д. Выбирая в ресторане комплексный обед, посетитель рассчитывает на его полноту, сбалансированность, разумную стоимость и определенное удобство, отдавая, впрочем, себе отчет, что ожидать одинаково высокого качества от всех

составляющих не приходится. Но если хочется именно этого, тогда нужно листать меню, выбирая отдельные блюда, и быть готовым к тому, что заплатить придется больше.

Нет таких поставщиков комплексных ИТ-систем, которые бы обладали одинаково сильной компетенцией во всех областях корпоративной деятельности. Кому-то лучше удаются «первые блюда» (финансовый и материальный учет, продажи), кому-то - вторые (производственные модули), кому-то - десерт (управление персоналом, аналитика). Некоторые же «блюда», такие, как решения для документооборота или биллинга, вообще не входят в комплексные пакеты, поскольку «универсалам» нет никакого смысла соревноваться с подобными специализированными решениями.

Ведущие разработчики пакетов деловых приложений уже особо и не настаивают, чтобы предприятия автоматизировали все свои бизнес-процессы на одном ядре. SAP, например, давно уже сменила концепцию в пользу интеграционной платформы, на которой можно объединять в КИС разнородные приложения. Пожалуй, только Oracle продолжает упорствовать, рекламируя достоинства работы на решениях одного поставщика. ИТ-директора, как люди технические, тоже в большинстве своем сторонники подхода best of breed.

Ни одна тиражная комплексная система не способна идеально «сесть» на реальный бизнес. Поэтому при внедрении всегда возникает дилемма: перестраивать бизнес под систему или «натягивать» систему, переписывая ее, на бизнес. Кастомизация тиражного ИТ-решения - дело обычное, любой продукт предполагает настройку под пользователя. Вопрос в пределах допустимой степени кастомизации. Если переписывать приходится слишком много и глубоко, не лучше ли изменить вкусы, то есть перестроить сложившиеся на предприятии бизнес-процессы под систему?

Исследования на западном корпоративном рынке показали, что удовлетворительной считается кастомизация на уровне 40% изменений. При этом кастомизацией западные ИТ-директора называют изменения в настройках

системы. В России же, как правило, это «глубокая» переделка системы под «настройки» бизнеса, далеко не всегда оправданная. Если такая переделка переваливает за уровень 50%, переделывать систему не имеет смысла: проблема уже в бизнесе».

В России уже есть примеры автоматизации бизнеса, создающегося «с нуля», когда внедрение ИТ-системы заложено в бизнес-план и автоматизация происходит параллельно с организацией бизнес-процессов и наймом персонала. Такие проекты реализуются в рекордно короткие сроки. Но если предприятие насчитывает не один десяток, а то и сотню лет, то провести изменения в таком сложившемся организме бывает очень непросто.

Хорошая стратегия - создание «вертикальных» комплексных решений, учитывающих отраслевую специфику бизнеса. Ведущие вендоры создают такие решения самостоятельно или призывают к работе на «последней миле» своих партнеров. И рынок все больше склоняется к отраслевым решениям, но отраслевой расклад - неравномерный, определяется спросом и готовностью оплачивать отраслевую кастомизацию. Для таких зрелых рынков, как розничная торговля, нефтедобыча и нефтепереработка, металлургия, торговля автомобилями отраслевые решения уже появились. Их создание оплачивают лидеры отрасли. А вот для транспортной, строительной индустрии специализированных решений нет.

Крайняя степень кастомизации - самостоятельно написать уникальную систему, которая будет учитывать все нюансы бизнеса. Надо сказать, что такой вариант автоматизации довольно популярен у ИТ-директоров российских предприятий (рисунок 2). Еще несколько лет назад вполне естественным такой выбор делало соотношение между стоимостью тиражных решений (включая затраты на внедрение) и зарплатой штатных программистов. Но многие предприятия начинают отказываться от самописных решений, понимая, что соревноваться с профессиональными командами, поставившими разработку софта на промышленную основу, в большинстве случаев бесполезно. За

разработку собственной системы сейчас берутся лишь тогда, когда необходимость более серьезной автоматизации какого-либо бизнес-процесса уже назрела, а подходящего готового решения, учитывающего всю необходимую специфику и проверенного на других российских предприятиях, ИТ-директор найти не может.

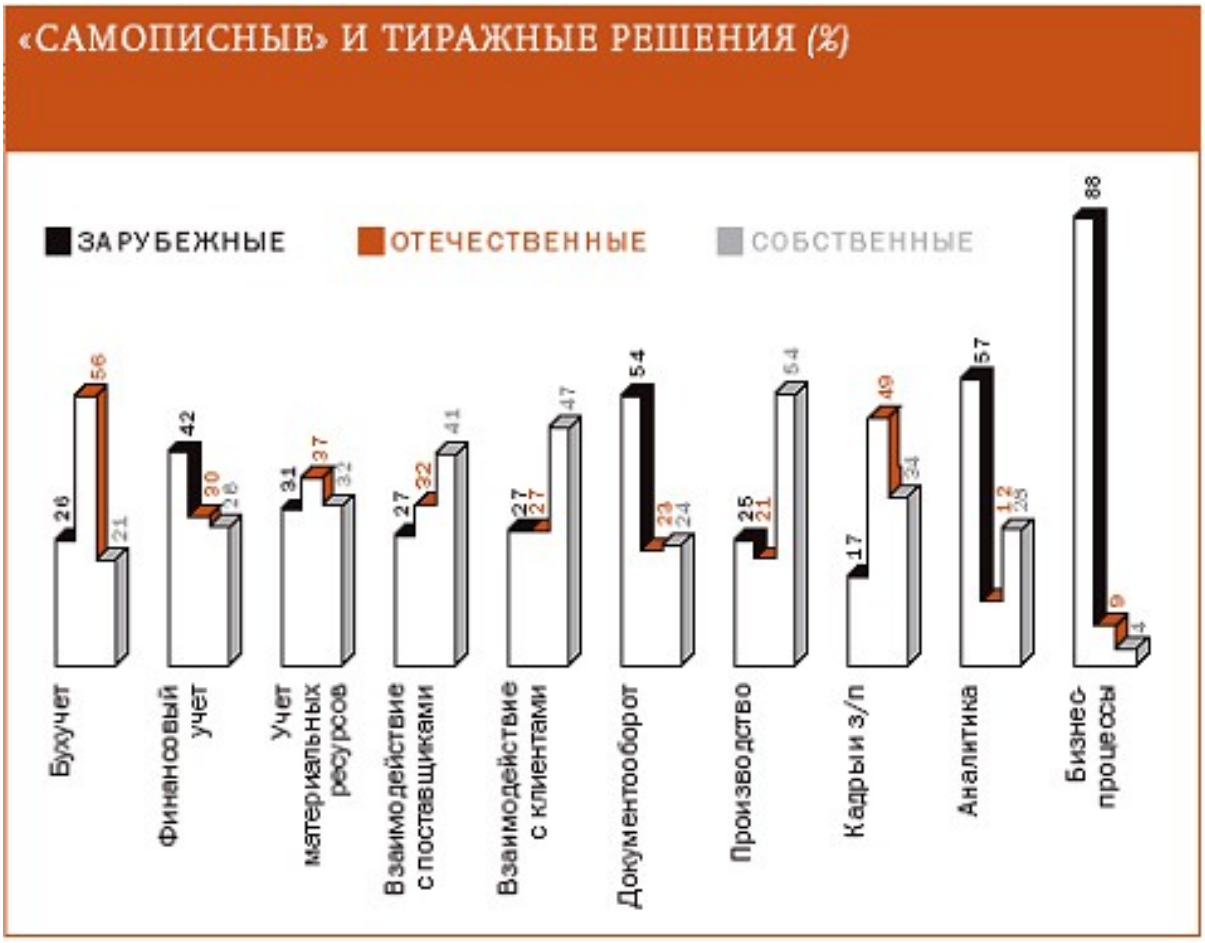

Рисунок 2 – Доля собственных интеграционных решений

Проведя исследование среди существующих автоматизированных информационных систем, можно сделать следующие выводы:

- существующие системы очень громоздки и рассчитаны на автоматизацию предприятия, а не отдельного бизнес-процесса;
- такие системы не отвечают необходимым требованиям и не подходят под структуру существующего процесса;

- стоимость приобретения таких систем, их интеграция и затраты на обучение персонала превышает затраты на самостоятельную разработку и внедрение. Также необходимо учесть затраты на техническую поддержку системы и кастомизацию;
- для собственной разработки требуемой АИС предприятие не имеет квалифицированного персонала.

Исходя из этих утверждений и описания предметной области можно заключить, что требуется разработка новой АИС, пригодной к внедрению в подобной ситуаций, т.к. требуемого программного средства на рынке ИТ нет. Такая АИС будет полностью отвечать требованиям поставленной задачи, полностью соответствовать бизнес процессам протекающим на предприятии, будет иметь большой потенциал дальнейшего развития автоматизации систем учёта, электронного документа оборота и планирования на данном предприятии. Дальнейшее развитие разрабатываемой АИС может дать ряд мощных функциональных средств решения целого класса подобных задач.

3. Инструментарий для построения требуемой АИС учёта

Существует несколько типовых технологий, на базе которых строятся системы информационного управления, таких как:

- технологии, основанные на использовании объекто-ориентированных визуальных сред разработки программного обеспечения и встроенных средств управления базами данных (Borland Delphi, Visual C++, Visual Foxpro и т.д.). Данный способ построения системы является зависимым от конкретной платформы разработки, что ограничивает возможности и универсализм в использовании системы, так как обычно приходиться устанавливать дополнительное ПО;
- Веб-технологии, основанные на использовании гипертекстовых средств с применением «скриптовых» языков (PHP, Perl, ASP, Phyton) в привязке с «сетевыми» СУБД (MySQL, Oracle, PostgreSQL, MS SQL). Этот способ разработки является более универсальным, так как доступ к системе возможен с помощью обычного Веб-браузера;
- технологии, основанные на Веб-сервисах - Сервис-ориентированная архитектура (англ. SOA, service-oriented architecture) — модульный подход к разработке программного обеспечения. В основе SOA лежат принципы многократного использования функциональных элементов ИТ, ликвидации дублирования функциональности в ПО, унификации типовых операционных процессов, обеспечения перевода операционной модели компании на централизованные процессы и функциональную организацию на основе промышленной платформы интеграции.

Рассмотрим подробнее каждый из способов реализации и инструменты, при помощи которых можно воплотить в жизнь то, разрабатываемую АИС.

3.1 Выбор архитектуры ИС

При проектировании клиент-серверных приложений необходимо определить, какие задачи будут выполняться на клиентском рабочем месте, а какие на сервере. Это решение влияет на стоимость клиентских и серверных частей системы, устойчивость и безопасность приложения в целом, гибкость разработки для дальнейшей модификации и адаптации [2].

Архитектура "Клиент-сервер". Клиент-сервер - это не только архитектура, это - новая парадигма, пришедшая на смену устаревшим концепциям. Суть ее заключается в том, что клиент (исполняемый модуль) запрашивает те или иные сервисы в соответствии с определенным протоколом обмена данными. При этом, в отличие от ситуации с файловым сервером, нет необходимости в использовании прямых путей операционной системы: клиент их "не знает", ему "известны" лишь имя источника данных и другие специальные сведения, используемые для авторизации клиента на сервере. Сервер, который физически может находиться на том же компьютере, а может - на другом конце земного шара, обрабатывает запрос клиента и, произведя соответствующие манипуляции с данными, передает клиенту запрашиваемую порцию данных, как это показано на рисунке 3.

В рамках направления "клиент-сервер" существуют два основных "диалекта": «тонкий» и «толстый» клиент.

В системах на основе тонкого клиента используется мощный сервер баз данных, это - высокопроизводительный компьютер и библиотека так называемых хранимых процедур, позволяющих производить вычисления, реализующие основную логику обработки данных, непосредственно на сервере. Клиентское приложение, соответственно, предъявляет невысокие требования к аппаратному обеспечению рабочей станции.

Преимущества «тонких» клиентов [3]:

- меньшие затраты на администрирование. ПО тонкого клиента практически полностью управляется с сервера. Кроме того, аппаратные средства имеют меньше возможностей для отказа из-за своей простоты, а ПО меняется крайне редко, что позволяет обеспечить защиту от вредоносных программ;
- легкость в обеспечении защиты информации. ПО тонкого клиента может быть спроектировано так, чтобы на клиенте не хранились данные. Тем самым обеспечивается централизованная защита от утечки информации и минимизируются риски физического повреждения данных;
- обеспечение непрерывности работы в случае аппаратных сбоев. При поломке оборудования тонкого клиента на время ремонта пользователю может быть быстро предоставлена полноценная замена, потому что все данные хранятся на сервере.

Системы с «толстым» клиентом, напротив, реализуют основную логику обработки на клиенте, а сервер представляет собой в чистом виде сервер баз данных, обеспечивающий исполнение только стандартизованных запросов на манипуляцию с данными (как правило - чтение, запись, модификацию данных в таблицах реляционной базы данных). В системах такого класса требования к рабочей станции выше, а к серверу - ниже.

Преимущества «толстых» клиентов:

- Не требуется высокопроизводительный сервер, так как клиент сам выполняет большую часть обработки данных. Это приводит к значительному удешевлению серверов;
- Работа в автономном режиме. Как правило, таким клиентам не требуется постоянное соединение с центральным сервером;
- Большая гибкость. Некоторые программные продукты проектируются для компьютеров, имеющих свои локальные ресурсы. Запуск

подобной программы на тонком клиенте может быть затруднителен или невозможен.

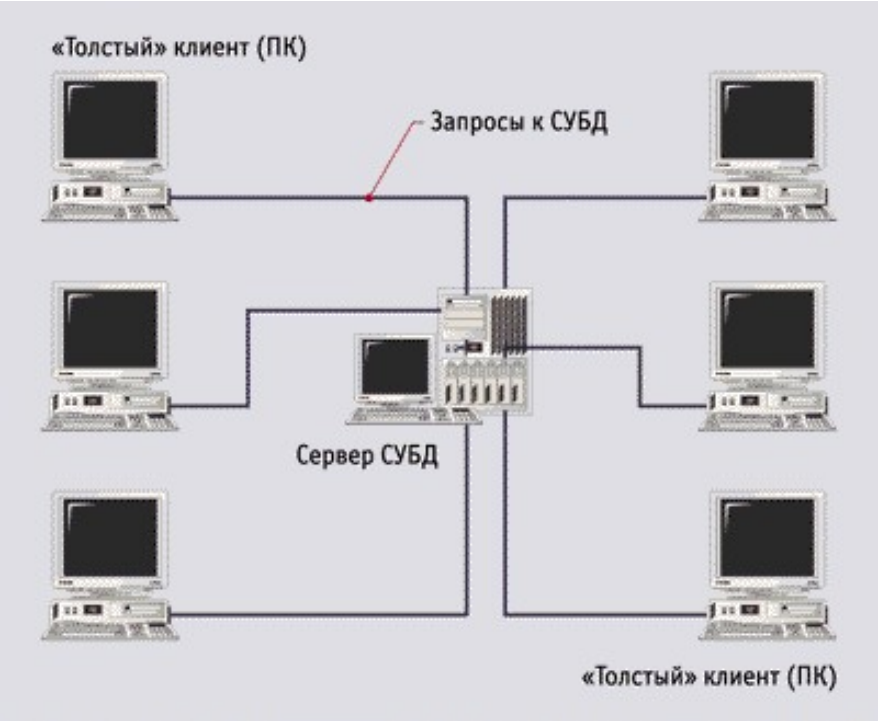

Рисунок 3 - Архитектура клиент-сервера

Достоинство архитектуры - переносимость серверной компоненты на серверы различных производителей: все промышленные серверы баз данных реляционного типа поддерживают работу со стандартизованным языком манипулирования данными SQL, но внутренний встроенный язык обработки данных, необходимый для реализации логики обработки на сервере у каждого из серверов свой [4].

Трехслойная архитектура. Базируется на дальнейшей специализации компонент - архитектуры: клиент занимается только организацией интерфейса с пользователем, сервер баз данных - только стандартизованной обработкой универсалам данных. Для реализации логики обработки данных архитектура предусматривает отдельный слой - слой бизнес-логики. Этот слой может представлять собой либо выделенный сервер (сервер приложений), либо размещаться на клиенте в качестве динамической библиотеки. Данная

архитектура позволила соединить достоинства тонкого и толстого клиентов: хорошая переносимость соединяется в ней с невысокими требованиями к клиенту [5].

С развитием интернет-технологий появилась разновидность трехслойной архитектуры на основании использования web-технологий [6] . В этой разновидности роль сервера приложений играет web-сервер, а в качестве клиента используется стандартный web-браузер. На рисунке 4 показана ситуация с использованием различных серверов для хранения баз данных и приложений, которые могут географически располагаться на разных концах земного шара и представлять собой не один десяток компьютеров. В моём случае веб-сервер и сервер баз данных могут располагаться на одном компьютере.

Достоинства - в пониженных требованиях к клиенту и в легкой встраиваемости данной архитектуры в мировые информационные сети.

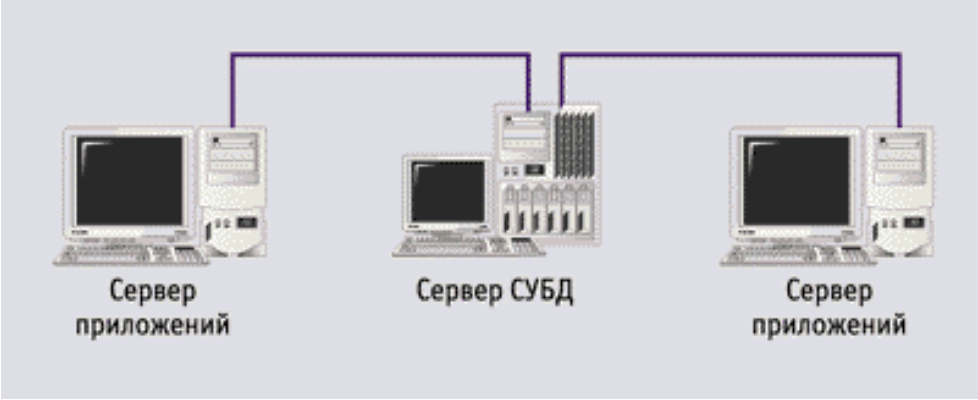

Рисунок 4 – Архитектура информационной системы на основе Web-среды

Основной недостаток - известные ограничения, накладываемые на интерфейс пользователя web-браузерами.

С помощью архитектуры SOA реализуются три аспекта ИТ-сервисов, каждый из которых способствует получению максимальной отдачи от ИТ в бизнесе [7]:

- сервисы бизнес-функций. Суть этих сервисов заключается в автоматизации компонентов конкретных бизнес-функций, необходимых потребителю;
- сервисы инфраструктуры. Данные сервисы выполняют проводящую функцию, посредством платформы, через которую поставляются сервисы бизнес-функций;
- сервисы жизненного цикла. Эти сервисы являются своего рода «обёрткой», которая в большинстве случаев поставляет ИТ-пользователям «настоящие сервисы». Сервисы жизненного цикла отвечают за дизайн, внедрение, управление, изменение сервисов инфраструктуры и бизнес-функций.

Ряд архитектурных особенностей SOA позволяет уменьшить степень связанности различных элементов системы. Для взаимодействия компонентов используется сравнительно небольшой набор простых интерфейсов, которые обладают только самой общей семантикой и доступны всем провайдерам и потребителям. Через эти интерфейсы (рисунок 5) передаются сообщения, ограниченные некоторым словарем. А поскольку даны только общая структура корпоративной системы и словарь, то вся семантика и бизнес-логика, специфичная для приложений, описывается непосредственно в этих сообщениях [8].

Корпоративная информационная система, построенная на основе SOA, состоит из набора сущностей, доступных через прикладные программные интерфейсы. Встроенный механизм поиска и обнаружения сервисов в общем реестре позволяет потребителю выйти на оператора, предлагающего искомую функцию.

Архитектура веб-сервисов также является сервисно-ориентированной. Более того, веб-сервисы — это суть SOA с двумя дополнительными ограничениями: интерфейсы базируются на интернет-протоколах (HTTP, FTP, SMTP, TCP), а все сообщения описываются в формате XML.

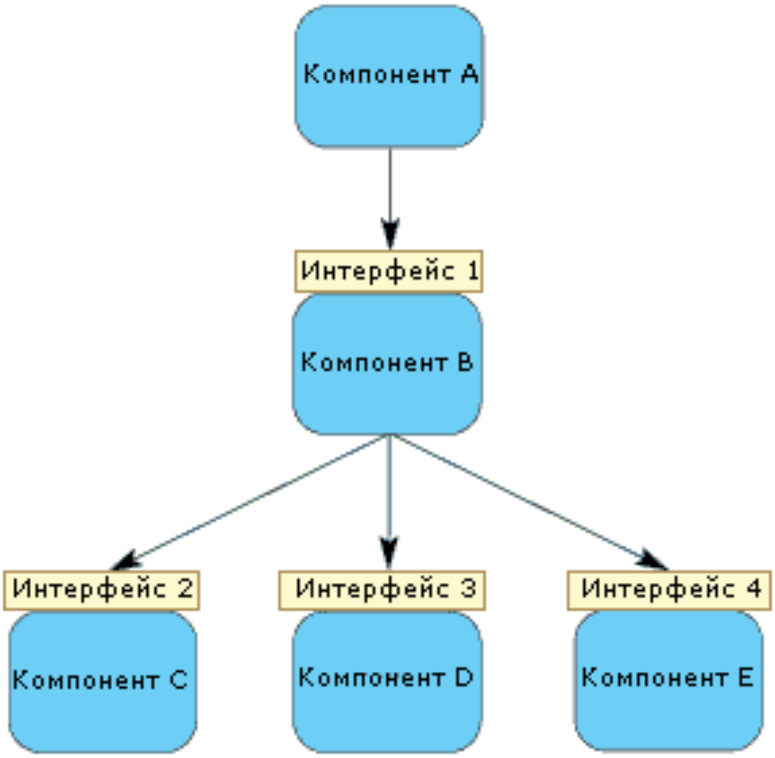

Рисунок 5 – Архитектура системы в концепции SOA

Практические аспекты сервисно-ориентированной технологии позволяют решить проблемы масштабируемости, интегрировать сети передачи данных и голоса, упростить процедуры проектирования и управления сетями, а также создать другие распределенные приложения, прозрачно взаимодействующие с ресурсами систем при помощи прикладных программных интерфейсов и открытых стандартов [9].

Вот шесть механизмов, с помощью которых поддерживается SOA-политика:

- операционная модель жизненного цикла SOA;
- организация SOA;
- SOA-процесс;
- портфель активов для сервисной интеграции в SOA;
- инструментарий SOA;
- SOA-технологии.

Эти механизмы используются обоими подходами к разработке и управлению SOA. Первый подход – это управление SOA по типу «сверху вниз». Он подразумевает, что управление по своей сути является стратегическим и начинается с модели и определённых проектов. Продвигаясь вниз, «стратегическое управление» определяет людей, процессы, сервисы, инструменты и технологии, которые будут привлекаться для поддержки корпоративного SOA-проекта. Второй подход – «снизу вверх» - соответственно подразумевает «тактическое управление», которое, наоборот, строит SOA-проект на основе создаваемых технологий, инструментов и сервисов. Большинство предприятий идет по пути «снизу вверх», начиная с конкретных сервис-ориентированных шагов, направленных на определённые предметные области. Очень редко встречаются организации, в которых создание стратегии первично по отношению к созданию необходимых отделов и бизнес-подразделений, первоначальных SOA-технологий и инструментария. Такой подход в целом только усложняет процесс налаживания управления SOA.

Автоматизированные учетные системы предприятия, как правило, являются многопользовательскими. Их архитектура позволяет различным пользователям со своих рабочих компьютеров одновременно получать доступ к информации, хранящейся в системе. Такие системы построены по технологии «клиент–сервер». В моём случае клиентами системы являются сотрудники отдела технической поддержки филиала предприятия и начальники информационно технических отделов его подразделений [10].

На данный момент развития коммуникационных технологий, единственным общедоступным средством для организации широкомасштабного удаленного информационного взаимодействия является сеть Internet, которая и была использована в качестве передающей среды для построения информационного обмена.

Поэтому, для решения задач автоматизации учета построение распределенной системы с ресурсоемким клиентским ПО не является

существенно необходимым. Гораздо более актуально снижение затрат на администрирование, обеспечение защиты информации, целостности данных и непрерывности работы. Существующие Web-сервисы не отвечают потребностям и целям, преследуемым предприятием, поэтому было принято решение ориентироваться на Web [11].

Опыт построения ИС по такой технологии широко применяется и занимает лидирующие позиции в России среди ИС. Примером реализации этого подхода является Naumen Service Desk — информационная система для ИТ-подразделений и сервисных компаний (http://www.naumen.ru/go/products /nausd). Она так же построена на основе трехзвенной архитектуры. Интерфейсы пользователя и администратора полностью web-ориентированы. Для работы пользователю необходим лишь браузер. На рисунке 6 эта архитектура показана применительно к разрабатываемой мной автоматизированной информационной системе управления исполнением заявок по ремонту оргтехники (АИС УИЗРО).

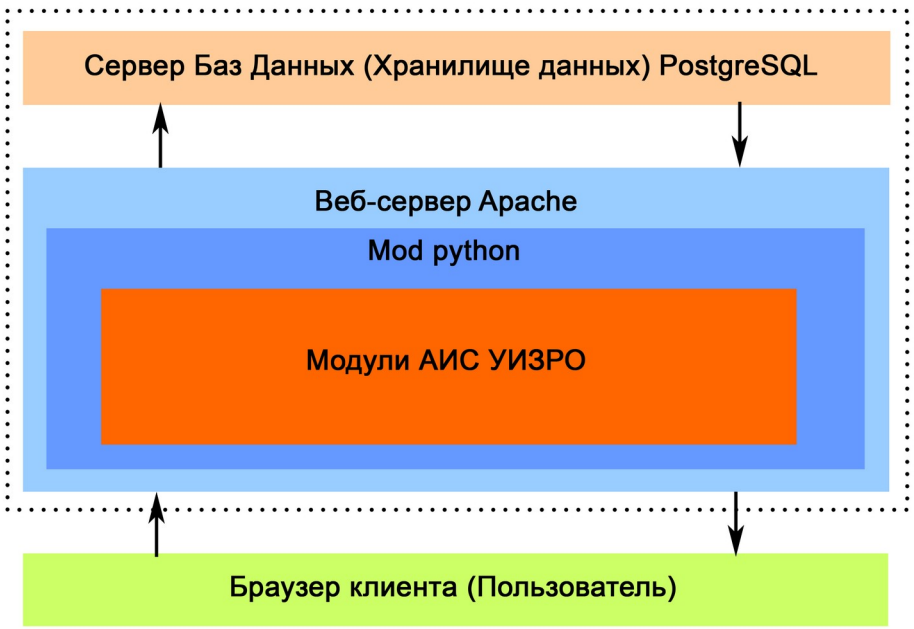

Рисунок 6 – Архитектура АИС УИЗРО

В качестве постоянного хранилища данных используется реляционная база данных. В базу данных помещаются только данные в виде набора

реляционных сущностей (связанных таблиц) без элементов программируемой логики (триггеров, хранимых процедур, представлений и т.п.).

Приложение имеет модульную архитектуру: функциональность системы сосредоточена в нескольких модулях, работающих независимо друг от друга и взаимодействующих через общую шину сервисов.

Рассмотрим более подробно инструментарий построения АИС.

3.2 Шаблон проектирования

Шаблоны проектирования (паттерн, англ. design pattern) — это многократно применяемая архитектурная конструкция, предоставляющая решение общей проблемы проектирования в рамках конкретного контекста и описывающая значимость этого решения [12]. Паттерн не является законченным образцом проекта, который может быть прямо преобразован в код, скорее это описание или образец для того, как решить задачу, таким образом, чтобы это можно было использовать в различных ситуациях.

Объектно-ориентированные шаблоны зачастую показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться. Алгоритмы не рассматриваются как шаблоны, так как они решают задачи вычисления, а не проектирования.

Главная польза каждого отдельного шаблона состоит в том, что он описывает решение целого класса абстрактных проблем. Также тот факт, что каждый шаблон имеет свое имя, облегчает дискуссию об абстрактных структурах данных (ADT) между разработчиками, так как они могут ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация терминологии, названий модулей и элементов проекта.

Правильно сформулированный паттерн проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. В отличие от идиом, шаблоны независимы от применяемого языка программирования.

При проектировании клиент-серверных приложений для ускорения разработки имеет смысл воспользоваться шаблонами проектирования, предоставляющими эффективные способы решения типовых задач создания компьютерных программ. Шаблон не является законченным образцом проекта, который может быть прямо преобразован в код, скорее, это описание или образец того, как решить задачу. Главное достоинство каждого шаблона состоит в том, что он описывает решение целого класса абстрактных проблем. За счет шаблонов производится унификация терминологии, названий модулей и элементов проекта. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова.

Model View Controller (MVC, управление видом модели) является одним из шаблонов архитектуры ПО. В сложных компьютерных приложениях, которые должны предоставлять пользователю большой объем данных, разработчик часто стремится разделить обработку данных и разработку пользовательского интерфейса так, чтобы изменения интерфейса не влияли на управление данными и данные могли быть реорганизованы без изменения интерфейса. Схема MVC решает эту проблему, отделяя доступ к данным и бизнес-логику от отображения данных и действий пользователя при помощи промежуточного компонента контроллера. Таким образом, модель данных приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на другие компоненты [13].

Модель обеспечивает хранение и представление информации, с которой работает приложение. Способ организации информации определяется особенностями предметной области. Многие приложения используют постоянный механизм хранения данных (такой, как БД). Однако в самом

шаблоне MVC не определяется способ доступа к данным, поскольку подразумевается, что он определяется в модели.

Вид, как правило, представляет собой интерфейс пользователя. Он выполняет функции представления данных модели в виде, удобном для обработки. Для одной модели может одновременно существовать несколько видов.

Контроллер обрабатывает данные, отвечает на события, как правило, пользовательские действия, и может вносить изменения в модель.

Хотя MVC реализуется по-разному, обработка данных обычно происходит следующим образом [14]:

- Пользователь производит некоторое действие в интерфейсе системы (например, нажимает кнопку или изменяет значение поля в форме);
- Контроллер обрабатывает событие, полученное от пользовательского интерфейса, часто посредством вызова определенного обработчика;
- Контроллер обращается к модели, обновляя данные в соответствии с действиями пользователя;
- Вид обновляет пользовательский интерфейс. Данные для этого вид получает из модели посредством контроллера;
- Пользовательский интерфейс ожидает дальнейших действий пользователя, которые запустят цикл снова.

Стандартную схему архитектуры «Модель-Вид-Контроллер» можно увидеть на рисунке 7. Важно отметить, что вид и контроллер зависят от модели. Однако модель не зависит ни от вида, ни от контроллера. Это одно из ключевых достоинств подобного разделения. Оно позволяет строить модель независимо от визуального представления. Разделение моделей и видов в MVC способствует упрощению проектирования архитектуры системы и повышению гибкости.

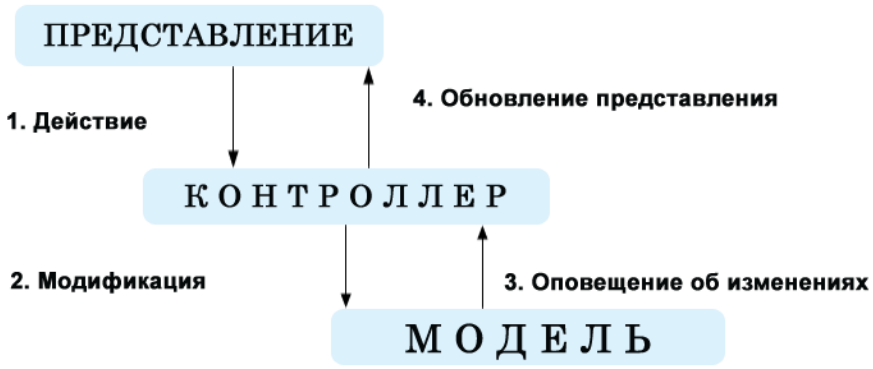

Рисунок 7 – Стандартная схема архитектуры «Модель-Вид-Контроллер»

Данная схема широко используется при построении web-приложений.

3.3 Реализация хранилища данных

Для хранения информации уже давно и успешно применяется реляционная модель данных. PostgreSQL — свободная объектно-реляционная система управления базами данных.

PostgreSQL является свободной альтернативой коммерческим СУБД (таким как Oracle Database, Microsoft SQL Server, IBM DB2, Informix и СУБД производства Sybase).

Она базируется на языке SQL и поддерживает многие из возможностей стандарта SQL:2003 (ISO/IEC 9075). Ограничения существующие в PostgreSQL на сегодняшний день приведены в таблице 1 [15].

Сильными сторонами PostgreSQL считаются:

- поддержка БД практически неограниченного размера;
- мощные и надёжные механизмы транзакций и репликации;
- наследование;
- легкая расширяемость.

Таблица 1 – Ограничения в PostgreSQL

Максимальный размер базы данных	Нет ограничений
Максимальный размер таблицы	32 ТБайт
Максимальный размер записи	1,6 ТБайт
Максимальный размер поля	1 ГБайт
Максимум записей в таблице	Нет ограничений
Максимум полей в таблице	250—1600, в зависимости от типов полей
Максимум индексов в таблице	Нет ограничений

Функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД. Хотя они могут писаться на чистом SQL, реализация дополнительной логики, например, условных переходов и циклов, выходит за рамки собственно SQL и требует использования некоторых языковых расширений. Функции могут писаться с использованием одного из следующих языков:

- встроенный процедурный язык PL/pgSQL, во многом аналогичный языку PL/SQL, используемому в СУБД Oracle;
- скриптовые языки — PL/Lua, PL/LOLCODE, PL/Perl, plPHP, PL/Python, PL/Ruby, PL/sh, PL/Tcl и PL/Scheme;
- классические языки — C, C++, Java (через модуль PL/Java);
- статистический язык R (через модуль PL/R).

PostgreSQL допускает использование функций, возвращающих набор записей, который далее можно использовать так же, как и результат выполнения обычного запроса.

Функции могут выполняться как с правами их создателя, так и с правами текущего пользователя.

Иногда функции отождествляются с хранимыми процедурами, однако между этими понятиями есть различие.

Триггеры определяются как функции, инициируемые DML—операциями. Например, операция INSERT может запускать триггер, проверяющий добавленную запись на соответствия определённым условиям. При написании функций для триггеров могут использоваться различные языки программирования.

Триггеры ассоциируются с таблицами. Множественные триггеры выполняются в алфавитном порядке.

Механизм правил (rules) представляет собой механизм создания пользовательских обработчиков не только DML-операций, но и операции выборки. Основное отличие от механизма триггеров заключается в том, что правила срабатывают на этапе разбора запроса, до выбора оптимального плана выполнения и самого процесса выполнения. Правила позволяют переопределять поведение системы при выполнении SQL-операции к таблице. Хорошим примером является реализация механизма представлений (англ. views): при создании представления создается правило, которое определяет, что вместо выполнения операции выборки к представлению система должна выполнять операцию выборки к базовой таблице/таблицам с учетом условий выборки, лежащих в основе определения представления. Для создания представлений, поддерживающих операции обновления, правила для операций вставки, изменения и удаления строк должны быть определены пользователем.

В PostgreSQL имеется поддержка индексов следующих типов: B-дерево, хэш, R-дерево, GiST, GIN. При необходимости можно создавать новые типы индексов, хотя это далеко не тривиальный процесс. Индексы в PostgreSQL обладают следующими свойствами:

- возможен просмотр индекса не только в прямом, но и в обратном порядке — создание отдельного индекса для работы конструкции ORDER BY ... DESC не нужно;
- возможно создание индекса над несколькими столбцами таблицы, в том числе над столбцами различных типов данных;

- индексы могут быть функциональными, то есть строиться не на базе набора значений некоего столбца/столбцов, а на базе набора значений функции от набора значений;
- индексы могут быть частичными, то есть строиться только по части таблицы (по некоторой её проекции); в некоторых случаях это помогает создавать намного более компактные индексы или достигать улучшения производительности за счет использования разных типов индексов для разных (например, с точки зрения частоты обновления) частей таблицы;
- планировщик запросов может использовать несколько индексов одновременно для выполнения сложных запросов.

PostgreSQL поддерживает одновременную модификацию БД несколькими пользователями с помощью механизма Multiversion Concurrency Control (MVCC). Благодаря этому практически отпадает нужда в блокировках чтения, и принцип ACID соблюдается очень эффективным способом.

PostgreSQL поддерживает большой набор встроенных типов данных:

- численные типы;
- целые;
- с фиксированной точкой;
- с плавающей точкой;
- денежный тип (отличается специальным форматом вывода, а в остальном аналогичен числам с фиксированной точкой с двумя знаками после запятой);
- символьные типы произвольной длины;
- двоичные типы (включая BLOB);
- типы «дата/время» (полностью поддерживающие различные форматы, точность, форматы вывода, включая последние изменения в часовых поясах);
- булев тип;

- перечисление;
- геометрические примитивы;
- сетевые типы;
- IP и IPv6—адреса;
- CIDR—формат;
- MAC—адрес;
- UUID—идентификатор;
- XML—данные;
- массивы;
- OID—типы;
- псевдотипы.

Более того, пользователь может самостоятельно создавать новые требуемые ему типы и программировать для них механизмы индексирования с помощью GiST.

PostgreSQL может быть расширен пользователем для собственных нужд практически в любом аспекте. Есть возможность добавлять собственные:

- преобразования типов;
- типы данных;
- домены (пользовательские типы с изначально наложенными ограничениями);
- функции (включая агрегатные);
- индексы;
- операторы (включая переопределение уже существующих);
- процедурные языки.

Таблицы могут наследовать характеристики и наборы полей от других таблиц (родительских). При этом данные, добавленные в порождённую таблицу, автоматически будут участвовать (если это не указано отдельно) в запросах к родительской таблице.

Данный функционал в текущее время не является полностью завершённым. Однако он достаточен для практического использования.

Прочие возможности:

- соблюдение принципов ACID;
- соответствие стандартам ANSI SQL-92 и SQL-99;
- поддержка запросов с OUTER JOIN, UNION, UNION ALL, EXCEPT и подзапросов;
- последовательности;
- контроль целостности;
- репликация;
- поддержка Юникода;
- поддержка регулярных выражений в стиле Perl;
- встроенная поддержка SSL и Kerberos;
- блокировки точнее, чем на уровне записи;
- подгружаемые расширения, поддерживающие SHA1, MD5, XML и другую функциональность (API открыт);
- средства для генерации совместимого с другими системами SQL-кода и импорта из других систем.

Согласно результатам автоматизированного исследования различного ПО на предмет ошибок, в исходном коде PostgreSQL было найдено 20 проблемных мест на 775 000 строк исходного кода (в среднем, одна ошибка на 39 000 строк кода). Для сравнения: MySQL — 97 проблем, одна ошибка на 4 000 строк кода; FreeBSD (целиком) — 306 проблем, одна ошибка на 4 000 строк кода; Linux (только ядро) — 950 проблем, одна ошибка на 10 000 строк кода.

Исходя из описанных выше характеристик [16] в качестве сервера баз данных я решил использовать PostgreSQL.

Использование реляционной модели для хранения информации об объектах реального мира часто обязывает приложение уметь обрабатывать данные в объектно-ориентированном виде, но при этом хранить их в

реляционной форме. Эта постоянная необходимость в преобразовании между двумя разными формами информации существенно снижает производительность и создает трудности для программистов, поскольку обе формы данных накладывают ограничения друг на друга [17].

Реляционные БД используют набор таблиц, представляющих данные простых типов. Дополнительная или связанная информация хранится в других таблицах. Часто для хранения одного объекта в реляционной БД используется несколько таблиц, что требует применения сложных запросов для получения всей относящейся к объекту информации и ее обработки. В результате увеличивается сложность программного кода, в нем легче допустить ошибку. Одним из решений данной проблемы является ORM (Object-relational mapping) – технология программирования, связывающая БД с концепциями объектно-ориентированных языков программирования, создавая виртуальную объектную БД.

С применением ORM система с точки зрения программиста выглядит как постоянное хранилище объектов, позволяя создавать объекты и работать с ними, автоматически сохраняя их в реляционной БД.

На практике возникают сложности следующего характера: слой транзакций, реализованный в ORM, может быть медленным и неэффективным. В результате программы работают медленнее и используют больше памяти, чем программы, написанные непосредственно на SQL.

Однако ORM избавляет программиста от написания большого количества кодов, тем самым значительно повышая его производительность [18]. Кроме того, большинство современных реализаций ORM позволяют программисту при необходимости самому жестко задать код SQL-запросов, который будет использоваться при тех или иных действиях (сохранение в БД, загрузка, поиск) с постоянным объектом.

3.4 Каркас приложения

Application Framework — каркас приложения (открытая инфраструктура приложения). Это программный framework, который используется, чтобы обеспечивать выполнение стандартной структуры приложения для определённой операционной системы. Каркасы приложения стали популярны с появлением GUI(Graphical user interface - графический пользовательский интерфейс (ГПИ), который имел тенденцию к распространению стандартой структуры для приложений. С их использованием стало гораздо проще создавать средства для автоматического создания ГПИ, так как структура внутренней реализации кода приложения стала известна заранее. Для обеспечения каркаса обычно используются техники объектно-ориентированного программирования, например части приложения могут наследоваться от базовых классов фреймворка [19].

Существует ряд некоммерческих продуктов с открытым программным кодом, которые можно использовать в качестве каркаса для построения web-приложений. Их использование позволяет программисту не заниматься низкоуровневым программированием. Применение таких средств делает код более простым и ясным, что способствует уменьшению количества ошибок в программе и сокращает время на разработку.

После изучения и сравнения таких средств создания web-приложений было решено использовать ряд компонентов свободного кроссплатформенного программного каркаса Django [20], основыванного на технологии ORM и реализующего архитектуру MVC(MTV). Он построен на языке Python, который широко используется при разработке web-приложений благодаря своей гибкости и универсальности. Django проектировался для работы под управлением Apache с модулем mod python и с использованием PostgreSQL в качестве базы данных.

Некоторые возможности Django:

- ORM, API доступа к БД с поддержкой транзакций;
- встроенный интерфейс администратора, с уже имеющимися переводами на многие языки;
- диспетчер URL на основе регулярных выражений;
- расширяемая система шаблонов с тегами и наследованием;
- система кеширования;
- интернационализация;
- подключаемая архитектура приложений, которые можно устанавливать на любые Django-сайты;
- «generic views» — шаблоны функций контроллеров;
- авторизация и аутентификация, подключение внешних модулей аутентификации: LDAP, OpenID и проч;
- система фильтров («middleware») для построения дополнительных обработчиков запросов, как например включённые в дистрибутив фильтры для кеширования, сжатия, нормализации URL и поддержки анонимных сессий;
- библиотека для работы с формами (наследование, построение форм по существующей модели БД);
- встроенная автоматическая документация по тегам шаблонов и моделям данных, доступная через административное приложение.

Различные компоненты фреймворка между собой связаны слабо, поэтому достаточно просто любую часть заменить на аналогичную.

С включением поддержки WSGI, Django может работать под управлением FastCGI, mod_wsgi или SCGI на Apache и других серверах (lighttpd, nginx,…) и подключаться к СУБД MySQL, SQLite и Oracle.

В составе Django присутствует собственный веб-сервер для разработки [21]. Сервер автоматически определяет изменения в файлах исходного кода проекта и перезапускается, что очень ускоряет процесс разработки на Python.

Но при этом он работает в одно-потоковом режиме и пригоден только для процесса разработки и отладки приложения.

Логика инструментария сосредоточена в модуле управления. Она включает в себя гибкие средства построения пользовательского интерфейса, отражающего структуру данных, возможность реализации бизнес-логики средствами инструментария без изменения базовых механизмов работы системы, механизмы генерации отчетов и контроля доступа к данным.

Логика приложения может быть распределена между контроллером, моделью и видом различными способами. Если требуется обеспечить наибольшую гибкость и универсальность, есть смысл придерживаться подхода, при котором бизнес-логика и семантика прикладной системы помещены в БД. Такой подход (Database-centric architecture) широко применяется при разработке web-приложений с использованием динамических языков программирования. Его преимущества [21]:

- система требует заметно меньшего взаимодействия между БД и сервером приложения, что уменьшает загрузку на сервер и время отклика приложения;
- операции внутри БД выполняются более эффективно, когда логика приложения описана в триггерах и хранимых процедурах;
- уменьшается количество кода, и он становится более эффективным из-за того, что программисту БД доступно много дополнительных возможностей по обработке данных.

Для реализации подобной архитектуры требуется описать в БД семантику и логику прикладной программы. В созданном инструментарии содержатся средства семантической разметки структуры БД и средства вызова хранимых процедур БД из пользовательского интерфейса.

Django, объеденяя в себе все возможности ORM и MVC, а так же за счёт своей модульности с лёгкостью позволяет решить поставленную передо мной задачу.

Django реализация ORM имеет возможность анализировать БД при ее подключении. При этом автоматически определяются состав и атрибуты полей таблиц, а также установленные связи между таблицами. Это дает возможность создания приложения, автоматически настраивающегося согласно декларированной структуре БД. Всю необходимую дополнительную информацию, описывающую семантику объектов и их связей, в том числе права доступа к данным, можно хранить либо в полях комментариев, либо в служебных таблицах БД.

При помощи этого механизма реализовано управление следующими возможностями: режим отображения таблицы (список, поиск значений, последние записи), видимость колонки, режим редактирования колонки, сортировка записей, расчет итогов по числовым колонкам, режим отображения связей между таблицами, ограничение выбора возможных значений поля.

Логику приложения в БД можно реализовать при помощи триггеров и хранимых процедур. Преимущество триггеров в том, что их вызов привязан к действию с данными, поэтому не требуется создание дополнительного механизма их вызова. Однако функциональность триггеров в MySQL ограничена, в частности, в триггерах запрещена рекурсия, что может стать серьезной проблемой при разработке эффективного прикладного решения. Поэтому был создан механизм вызова хранимых процедур БД. При операции с данными проверяется наличие в БД соответствующих процедур и вызывается их выполнение. Это позволило реализовать анализ и обработку вводимых данных, поддержание логической целостности информации и другие необходимые функции.

3.5 Web-сервер

Apache - свободный веб-сервер номер один в Интернет (http://www.netcraft.co.uk/survey/), где более половины всех интернет сайтов

работают под Apache или производных от него [22]. По статистике Netcraft, в августе 2007 года он работал на 51 % всех веб-серверов, в марте 2009 года — на 49%. На сегодняшний день Apache поддерживает функциональность приблизительно 19 млн. Веб-серверов.

Преимущества Apache заключаются в модульной организации, стабильности и быстродействии. При наличии соответствующих аппаратных средств и настройке он может выдерживать самые высокие загрузки. Вот не большой список ресурсов основанных на различных версиях этого сервера [23]:

- Yahoo;
- Altavista;
- GeoCities;
- Hotmail.

Веб-сервер Apache разрабатывается и поддерживается открытым сообществом разработчиков под эгидой Apache Software Foundation (http://www.apache.org/) и включён во многие программные продукты, среди которых СУБД Oracle и IBM WebSphere.

Причины успеха Apache:

1) Технологическое лидерство - в начальный период своей истории появления Apache был одним из технологических лидеров на рынке. Производительность была достаточно велика, потребности в ресурсах - малы. При этом программа обладала возможностью легкого расширения путем добавления модулей, причем реализовано это было на много лучше чем у конкурентов. Возможность легкого расширения обусловило появление большого количества производных от Apache серверов - Stronghold, Apache/SSL, mod_perl - удовлетворяющих большинство потребностей Веб-разработчиков по сегодняшний день.
2) Технологический консерватизм - Авторы популярных программ быстро оказываются заваленными пожеланиями пользователей. Если им следовать, то программы перегружаются функциональностью,

нужной только малому числу клиентов, сложность кода растет одновременно с числом проблем и т.д. Авторам Apache удалось сохранить необходимый баланс в этой области, разрабатываемое ими ПО имеет репутацию стабильного и предсказуемого.

3) Открытость процесса разработки. Процесс разработки Apache открыт для наблюдения и комментирования всем желающим и поэтому предсказуем. Это позволяет выпускать дополнительные модули к новым версиям практически одновременно с их выходом.
4) «Демократическая» разработка - проекте Apache реализована уникальная схема разработки - по каждому изменению проводится голосование, причем «существенные» изменения могут быть остановлены правом вето любого члена группы разработчиков, а несущественные -- должны набрать больше голосов «за», чем «против». Такая схема позволяет блокировать сомнительные технологические решения, поддерживая технологический консерватизм. Очевидно, что такая схема не может быть применена в произвольном проекте так как требует разумности от всех разработчиков группы, что достижимо далеко не всегда.
5) Поддержка пользователей - несмотря на огромную пользовательскую базу и некоммерческий статус, поддержка Apache была и остается очень хорошей по качеству - сообщения о проблемах анализируются в течение 1-2 дней.
6) Лицензирование - существенной причиной успеха Apache является действительно свободное лицензирование. Apache License, в отличие от наиболее распространенной в среде OpenSource GNU GPL, не навязывает свободное распространение производных работ, а требует лишь сохранения права на имя -- указания, что производный проект использует код, разработанный Apache Group. При такой схеме лицензирования коммерческие компании охотнее вкладывают свои

ресурсы в развитие продукта, примером может служить участие IBM в разработке Apache 2.0 и переносе Apache на платформу Windows.

Все перечисленные причины успеха Apache являются существенными, отсутствие любой из них заметно ухудшило бы продукт в глазах части, либо всех пользователей.

Единственным недостатком наиболее часто считается отсутствие удобного стандартного интерфейса для администратора [24].

Учитывая специфику поставленной задачи необходимо заключить, что для достижения целей поставленных в пункте 1.2, необходимо построить распределённую информационную систему с применением Веб-технологий.

Выбирая, рассмотренное ниже программное обеспечение и технологии я опирался на экономические выгоды, простоту и надёжность использования, скорость разработки и сопровождаемость используемых программных средств (ПС).

Информационная система должна иметь клиент-серверную архитектуру основанную на Веб-технологии с применением сервера баз данных, веб-сервера и стандартных средств представления веб-интерфейса, предоставляемых любым Web-браузером. Полученная таким образом система является кросс-платформенной, легко расширяемой и модернезируемой. Развёртывание и администрирование такой системы не требует дополнительных затрат на обучение персонала, или привлечение сторонних специалистов. Все описанные ниже ПС имеют свободную лицензию, легко доступны к скачиванию с официальных сайтов, локализованы и имеют огромное количество доступной справочной литературы, в том числе на Русском языке.

В случае, когда предприятие уже имеет работающий веб-сервер развёртывание такой системы сведётся к добавлению модуля соответствующего интерпретатора языка программирования к серверному ПО.

4 Моделирование АИС учёта и исполнения заявок ремонта оргтехники

AllFusion ERwin Data Modeler представляет собой набор средств концептуального моделирования данных, использующих метод IDEF1X. ERwin Data Modeler реализует проектирование схемы БД, генерацию ее описания на языке целевой СУБД (Oracle, Sybase, DB2, Microsoft SQL Server и др.) и реверсный инжиниринг существующей БД. ERwin Data Modeler выпускается в нескольких конфигурациях, ориентированных на наиболее распространенные средства разработки приложений.

AllFusion ERwin Process Modeler (BPwin) - средство моделирования бизнес-процессов, реализующее метод IDEF0, а также поддерживающее диаграммы потоков данных и IDEF3. В процессе моделирования BPwin позволяет переключиться с нотации IDEF0 на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. BPwin поддерживает функционально-стоимостной анализ (ABC) [25].

Учитывая особенности реализуемого проекта необходимо смоделировать деловые процессы протекающие при организации и планировании ремонтов и ТО от поломки и подачи заявки на ремонт до отгрузки отремонтированной техники и установки её на рабочее место. Для этих целей я воспользуюсь AllFusion ERwin Process Modeler. Это поможет определить, какую информацию необходимо учесть при создании базы данных и какие функции я смогу автоматизировать при помощи проектируемой АИС. Структуру самой базы данных я спроектирую в AllFusion ERwin Data Modeler, построив модель данных, учитывающую сущности, атрибуты и отношения между ними.

4.1 Моделирование в AllFusion ERwin Process Modeler

Основные причины функционального моделирования:

- реконструкция устройства систем (ФМ AS IS);

- поддержка анализа работы систем;
- ускорение и удешевление разработки систем (ФМ ТО ВЕ);
- компьютерная поддержка разработки программного обеспечения (осуществление CASE - технологии; совместно с другими программными средствами);
- подготовка проектного управления - разработка планов работ, основанных не на эвристике xix), а на функциональной модели деятельности, направленной на достижение той или другой цели;
- обоснование необходимости совершенствования функционирования (реинжиниринга) систем за счёт функционально - стоимостного анализа;
- упорядочение и накопление знаний из различных предметных областей;
- совершенствование текстов на естественном языке за счёт конвертации их в формат функциональной модели;
- реконструктивное моделирование (воссоздание системы функций).

На основе изложенных технических возможностей функционального моделирования можно указать на несколько наиболее известных и эффективных вариантов применения ФМ:

- поддержка перестройки работы предприятий; осуществляется совместно с другими методами и средствами;
- поддержка разработки систем менеджмента качества в соответствии с ISO 9000:2000;
- поддержка внедрения ERP – систем;
- создание систем поддержки жизненного цикла продукции (CALS);
- поддержка учебного процесса в различных отраслях знаний. Эффективность представления и усвоения знаний основывается на структурировании учебной информации и представлении её в графическом виде;

- создание инструктивных материалов, в том числе - должностных инструкций. Основывается на детальной проработке функционирования системы в формате ФМ;
- разработка компьютерноподдерживаемых планов деятельности, основанных на спецификации Activity (Activity Name Report), как составе работ плана;
- реконструктивное моделирование. Одна из возможных целей реконструктивного моделирования - поддержка работы исследователя, ведущего криминалистическое расследование.

Функциональная модель (ФМ) является средством описания работы (функционирования) той или другой системы или объекта. Может быть использована как для анализа работы существующих систем, так и для подготовки создания новых систем [26]. Практически без ограничений по предметной области.

Определение функциональной модели: "IDEF0 Модель: Графическое описание системы или объекта, которое разработано для определенной цели и с выбранной точки зрения. Набор из одной или большего количества IDEF0 диаграмм, которые изображают функции системы или предметной области в графике, тексте и глоссарии".

Функциональная модель разрабатывается как некоторый набор диаграмм, текстов и глоссария.

Каждая диаграмма разрабатывается как отдельный лист в формате от A4 до A1 или Customer (пользовательский) Основой функциональной модели является Activity - функция, графическим символом которой является прямоугольник (Box - в первоисточнике), в котором вписано имя функции в виде глагола или отглагольного существительного.

Каждая функция (прямоугольник Activity) встраивается в ФМ четырьмя типами связей - вход (Input) - всегда стрелка, символизирующая вход, направлена слева, в левую грань прямоугольника функции; выход (Output) -

всегда из правой грани прямоугольника направо; управление (Control) - всегда стрелка сверху вниз, на верхнюю грань прямоугольника; механизм (Mechanism) выполнения функции - стрелка снизу к нижней грани прямоугольника Activity, как показано на рисунке 8.

Это означает, что осуществление указанной в прямоугольнике функции приводит к преобразованию входа - в выход; что это преобразование осуществляется при помощи механизма и под управлением, символизированным нижней стрелкой.

Первая диаграмма ФМ называемая TOP (верхней или контекстной) - диаграммой, имеет только один прямоугольник Activity, который символизирует работу системы в целом. Все связи на этой диаграмме являются связями моделируемой системы со средой функционирования.

Каждая Activity, начиная с Activity TOP диаграммы, может быть декомпозирована (разделена) на субфункции, представляемые несколькими Activities, как это показано на рисунке 9.

Каждая Activity на диаграмме декомпозиции также оснащается теми же 4 видами связей, что и на TOP диаграмме; эти связи идентифицируются аббревиатурой ICOM, от Input, Control, Output, Mechanism.

Каждая Activity любой диаграммы обязательно должна иметь не менее одной стрелки управления и не менее одной стрелки выхода.

Оба конца любой стрелки должны быть привязаны (графически - касаться) либо к какой-либо границе диаграммы, либо к одной из сторон бокса Activity. Стрелки могут быть граничными - если они начинаются или оканчиваются на границе диаграммы, или внутридиаграммными, если они начинаются и заканчиваются на прямоугольниках Activities.

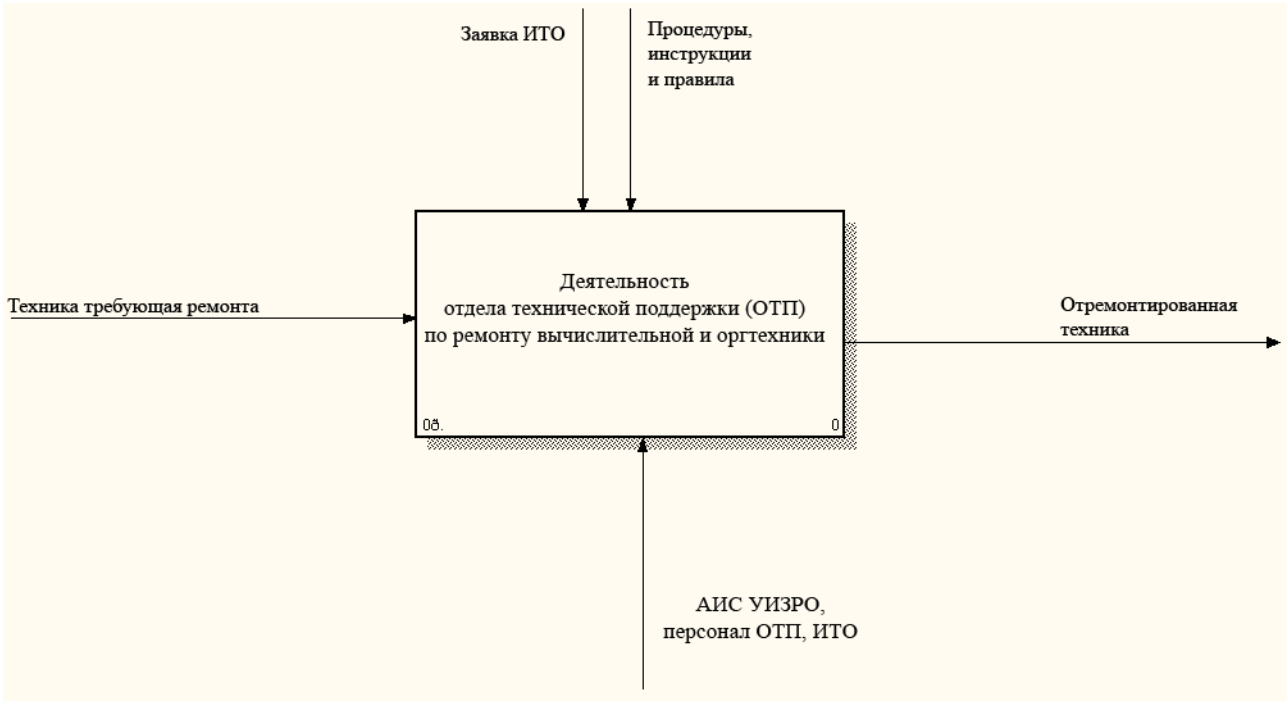

Рисунок 8 – Контекстная диаграмма (TOP-диаграмма) моделируемого процесса

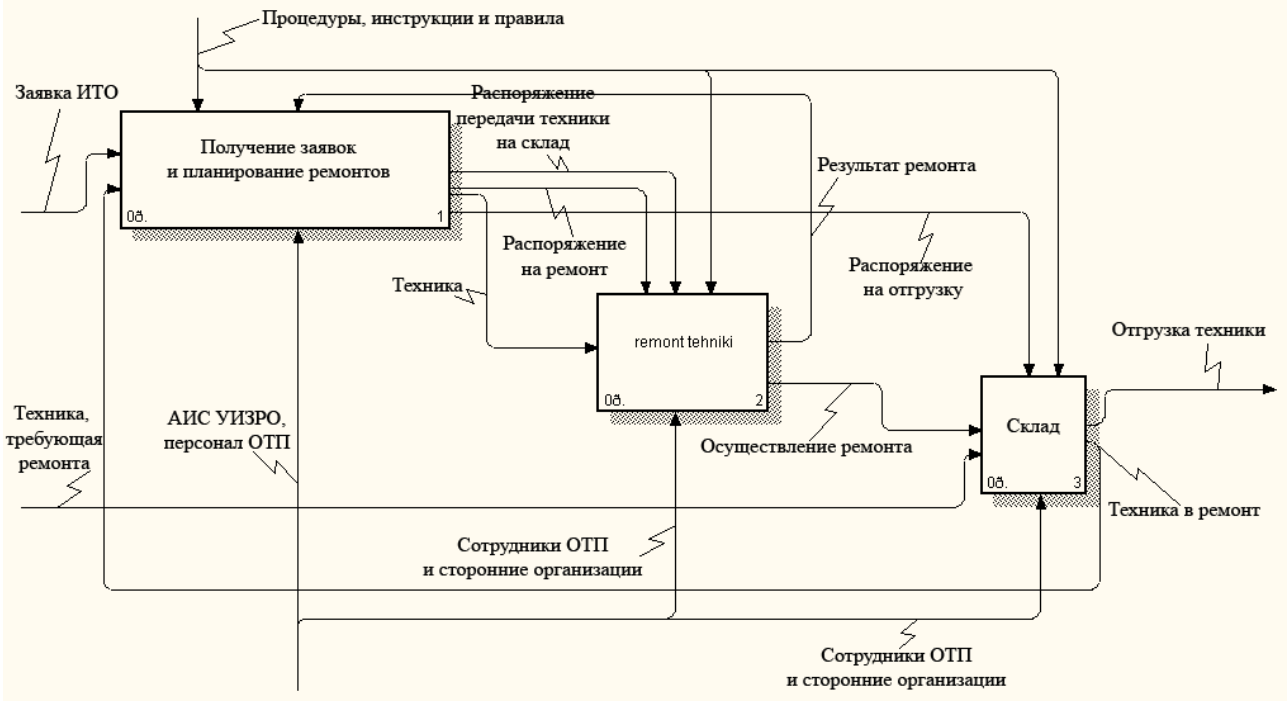

Рисунок 9 – Диаграмма декомпозиции A0, разработанная в методике IDEF0

На возникающую при формировании системы связей Activities диаграммы декомпозиции топологическую структуру IDEF0 накладывает ограничения, допускающие использование не всех связей, возможных геометрически.

Диаграммы декомпозиции называются дочерними диаграммами - для диаграммы, Activity которой подверглась декомпозиции. Эта диаграмма называется родительской. Разработка функциональной модели может быть произведена как ФМ AS IS - "функциональная модель как есть", или как ФМ TO BE - "функциональная модель как должно быть". Первая из них описывает функционирование существующей системы, вторая является проектным описанием работы создаваемой системы, или проектным описанием существующей системы, подвергающейся модернизации.

ФМ "как есть" является объектом и отчасти инструментом анализа, и позволяет выявить причины нерационального функционирования исследуемой моделируемой системы. ФМ "как должно быть" является, обычно, основой технического задания на создание или модернизацию системы.

Помимо IDEF0, информационные системы могут быть отмоделированы, в методике DFD, также поддерживаемой AllFusion Process Modeler, и отличающейся от IDEF0 рядом упрощений в её методических требованиях.

Информационные системы являются частью систем материально - информационных, их нижним иерархическим уровнем. Поэтому, после исчерпания описания материальных функций системы, дальнейшие декомпозиции могут осуществляться, в пределах одной комплексной совмещенной (гибридной) модели, в методике DFD.

При моделировании АИС я применил метод ФМ TO BE - "функциональная модель как должно быть". Полученные при этом ТОП диаграмму и её декомпозицию можно было видеть на рисунках 8 и 9 данной главы. Но для изучения предмета автоматизации необходимо более детально рассмотреть протекающие в системе процессы. Для этого я построю ещё ряд

диаграмм, являющихся декомпозициями функций, входящих в диаграмму декомпозиции. Рисунок 10 отображает, какие функции входят в такую функцию как «Получение заявок и планирование ремонтов».

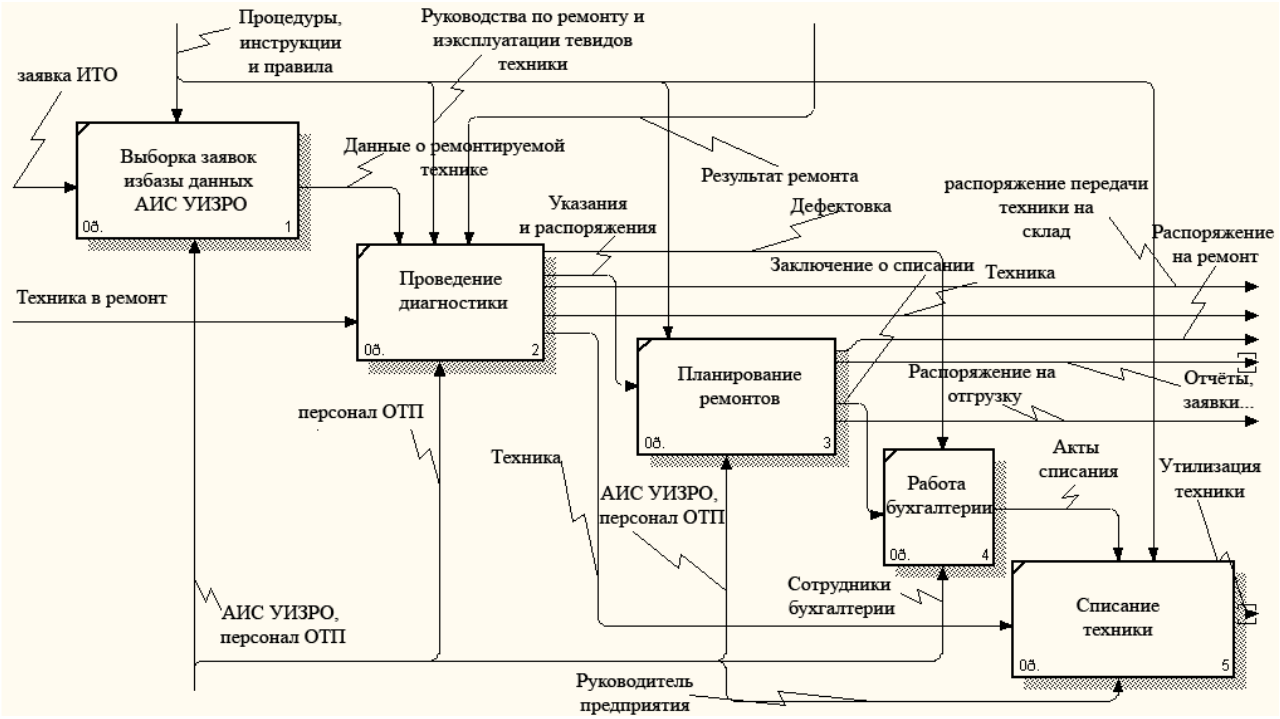

Рисунок 10 – Диаграмма декомпозиции А1. «Получение заявок и планирование ремонтов»

На рисунках 11 и 12, показаны декомпозиции таких функций как «Склад» и «Ремонт техники». Как видно из рисунков представленных диаграмм, что дальнейшей декомпозиции необходимо подвергнуть лишь ту часть системы, которая берёт на себя функции по планированию и администрированию всех процессов ОТП. В качестве родительской диаграммы я предлагаю использовать диаграмму декомпозиции А1, т.к. на ней содержатся функции, требующие автоматизации.

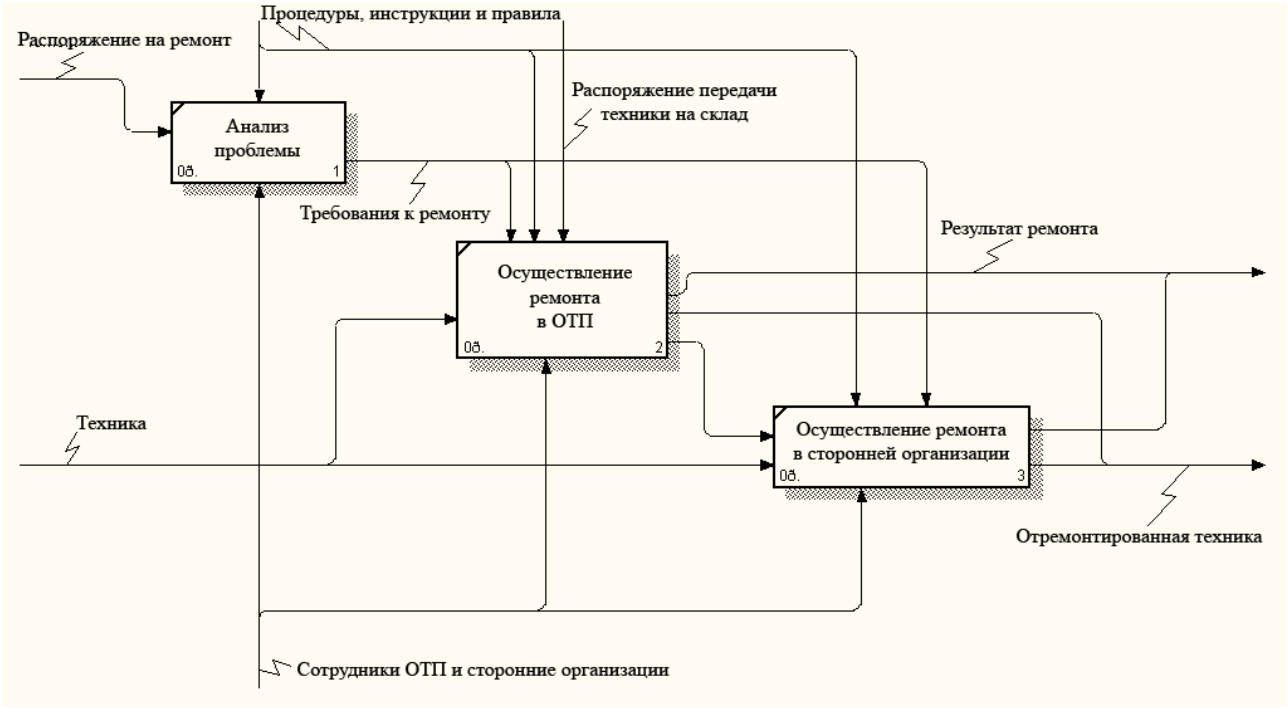

Рисунок 11 – Диаграмма декомпозиции А2 «Ремонт техники»

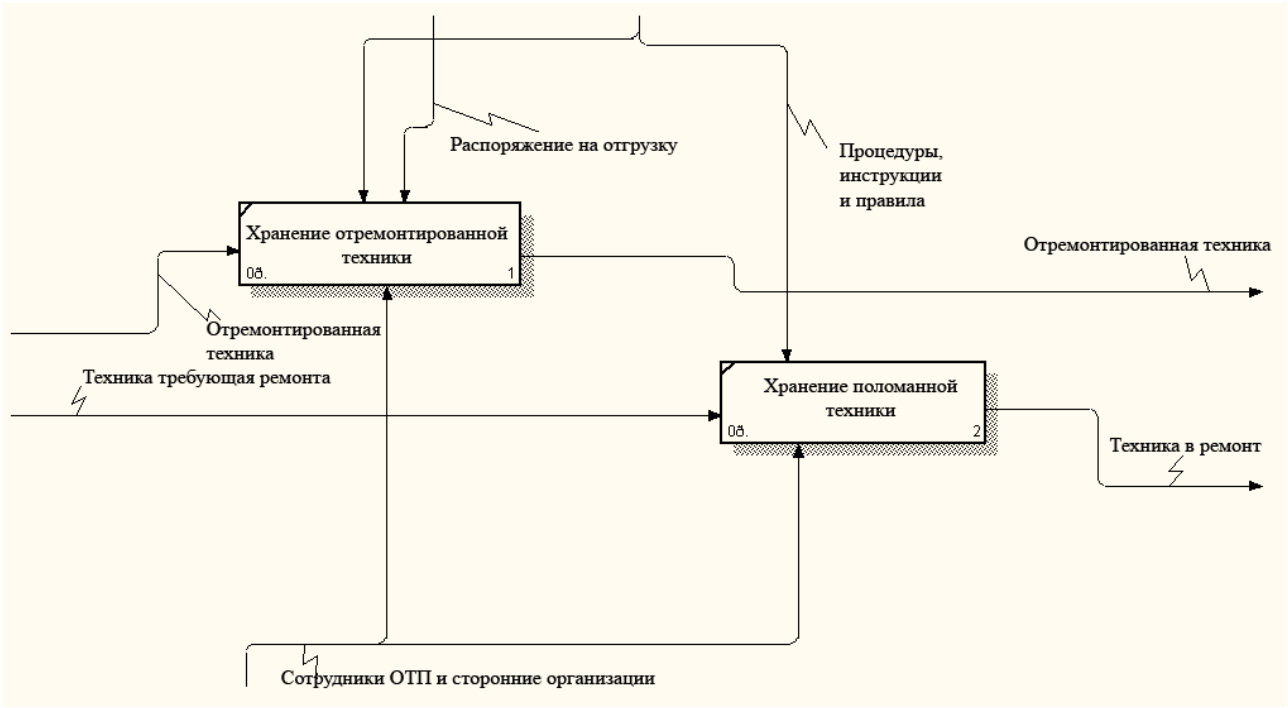

Рисунок 12 – Диаграмма декомпозиции А3 «Склад»

4.2 Моделирование в AllFusion ERwin Data Modeler

Обработка счетов. Электронная торговля. Анализ данных. Управление знаниями. Все это невозможно без использования баз данных. Системы с архитектурой клиент/сервер строятся на основе реляционных серверных СУБД. Приложения для Internet и интрасетей осуществляют доступ и динамическое обновление данных. Пакеты программ необходимо адаптировать и интегрировать с существующими системами. Хранилища данных объединяют и интегрируют множество баз данных, обеспечивая необходимые бизнесу гибкость и интеллектуальность. Успех применения всех этих приложений зависит от того, насколько хорошо спроектирована база данных.

ERwin Data Modeler - мощное и простое в использовании средство конструирования баз данных завоевавшее широкое признание и популярность. Оно обеспечивает высочайшую продуктивность труда при разработке и сопровождении приложений с использованием баз данных [27].

На протяжении всего процесса - от логического моделирования требований к информации и бизнес-правил, которые определяют базу данных, до оптимизации физической модели в соответствии с заданными характеристиками - ERwin позволяет наглядно отобразить структуру и основные элементы БД.

ERwin - это не просто мощное средство проектирования, но и инструмент разработки, способный автоматически создавать таблицы и генерировать тысячи строк текста хранимых процедур и триггеров для всех популярных СУБД. Революционная технология Complete-Compare (Завершить-Сравнить) позволяет организовать итеративную разработку, поддерживая постоянную согласованность модели и базы данных. Благодаря интеграции с популярными средами разработки программ, ERwin позволяет ускорить создание приложений для обработки данных.

Эта мощная система управления моделями позволяет проектировщикам баз данных разработчикам приложений и пользователям коллективно работать с информацией о моделях ERwin. Благодаря возможностям разбиения на фрагменты, а также совместного и многократного использования моделей, может быть повышена эффективность моделирования и обеспечено соблюдение корпоративных стандартов.

ERwin облегчает проектирование баз данных. Для этого достаточно создать графическую E-R модель (объект-отношение), удовлетворяющую всем требованиям к данным и ввести бизнес-правила для создания логической модели, которая отображает все элементы, атрибуты, отношения и группировки. Развитые средства моделирования помогают лучше спроектировать базу данных. Предусмотрены возможности манипулирования атрибутами путем их буксировки, внесения изменений и нормализации "на лету". Средства редактирования непосредственно на диаграммах позволяют вносить в модель изменения, не открывая специальных диалоговых окон. Навигация по отношениям обеспечивает быстрое перемещение в больших моделях для перехода к родительским или дочерним объектам. Формируемые системой отчеты позволяют быстро проверить корректность спроектированной базы данных.

Представления БД поддерживаются как интегрированные компоненты модели, что позволяет автоматически отображать в их описаниях изменения, внесенные в базовые таблицы. Автоматический перенос ключей обеспечивает ссылочную целостность базы данных.

Кроме того, ERwin позволяет работать с большими моделями общекорпоративного масштаба, разбивая их на фрагменты и легко управляемые подмножества, предоставляя отдельным специалистам возможность сосредоточить свои усилия в определенной области. Возможность сохранения отображений позволяет хранить множество представлений одной предметной области, ориентированных на различную целевую аудиторию.

Созданные с помощью ERwin модели данных можно редактировать, просматривать и распечатывать различными способами. В состав ERwin входит RPTwin - простая в использовании, оснащенная графическим интерфейсом утилита для формирования отчетов и встроенное средство для просмотра с настраиваемыми режимами, которые обеспечивают полный контроль над отображением содержимого отчетов. Кроме этого, уникальный интерфейс, построенный на использовании шаблонов, позволяет реализовать единые стандарты проектирования и отображать настройки для всех моделей.

ERwin устанавливает естественную динамическую связь между моделью и базой данных, что позволяет реализовать как прямой, так и обратный инжиниринг. Используя эту связь, ERwin автоматически генерирует таблицы, представления, индексы, правила поддержания целостности ссылок (первичных и внешних ключей), устанавливает значения по умолчанию и ограничения для доменов/столбцов. В состав ERwin включен целый ряд оптимизированных шаблонов триггеров, обеспечивающих целостность ссылок, и мощный макроязык, который позволяет создавать собственные триггеры и хранимые процедуры. Таким образом могут быть автоматически сформированы тысячи строк кода, что обеспечивает непревзойденную продуктивность разработки на основе моделей.

Средства расчета объема позволяют точно оценить первоначальный размер и характер роста базы данных или хранилища, облегчая эффективное распределение ресурсов системы и планирование мощности.

База данных может быть спроектирована и создана без написания отдельных SQL-предложений типа CREATE TABLE или INDEX. Поскольку физическая схема формируется на основе описательной логической модели, ваше приложение будет сразу же полностью документировано. ERwin позволяет также проводить обратный инжиниринг существующих баз данных путем построения модели непосредственно на основе ее таблиц. Таким образом

можно получить четкое представление о структуре и содержании существующего приложения.

Диаграмма сущность-связь (Entity Relationship Diagram (ERD)) включает сущности и взаимосвязи, отражающие основные бизнес-правила предметной области. Такая диаграмма не слишком детализирована, в неё включаются основные сущности и связи между ними, которые удовлетворяют основным требованиям предъявляемым к ИС. Диаграмма сущность-связь может включать связи «многие ко многим» и не включать описание ключей. Как правило ERD используется для презентации и обсуждения структуры данных с экспертами предметной области. Модель данных (DM) кроме описанного выше включает описание атрибутов сущностей – полей, входящих в базу данных таблиц [28].

На рисунке 13 представлена модель данных проектируемой АИС.

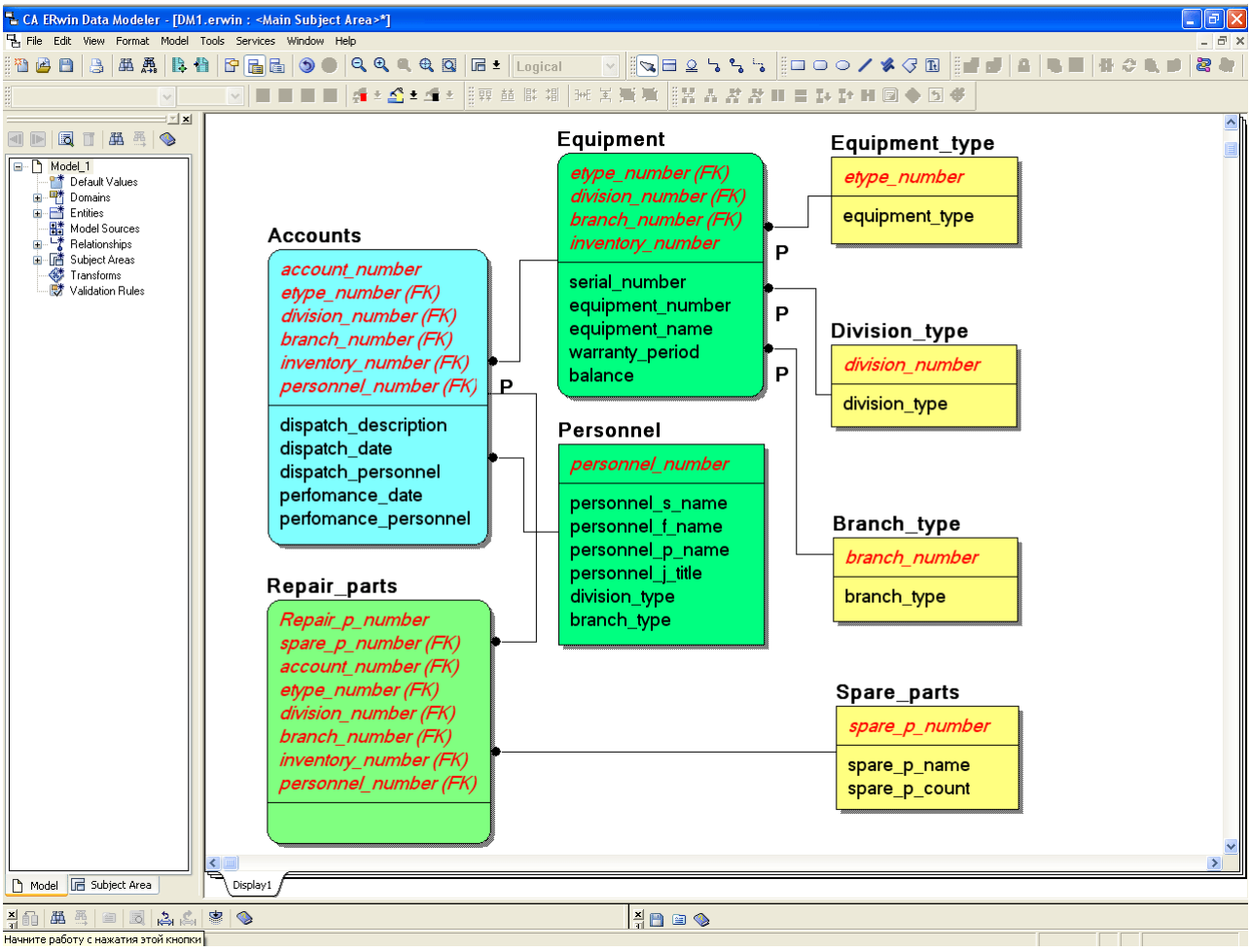

Рисунок 13 – модель данных в AllFusion ERwin Data Modeler

Из модели видно, что такая база данных может обеспечить минимальный набор возможного функционала системы. Она отвечает требованиям, выдвинутым в техническом задании и пригодна для тестирования и пробного запуска.

Используя различные SQL – запросы к БД мы можем получать всю необходимую информацию в соответствии с правами доступа, выполнять все стандартные для СУБД действия. База данных будет формироваться автоматически при подаче заявки и её исполнении.

4.3 Разработка интерфейса АИС УИЗРО

Использование в качестве архитектуры Веб-технологии предполагает наличие веб-интерфейса пользователя, предоставляемого браузером, что в свою очередь накладывает ряд ограничений, диктуемых применением HTML. Но, например, такие недостатки веб-интерфейса, как отсутствие динамичности и медленная работа на некоторых задачах ни как не скажется на работе с АИС. И напротив, такие достоинства для пользователя, как простота использования и простота разработки для разработчика скажутся на АИС положительно. Также отсутствует необходимость создания сложного дизайна, поэтому интерфейс разрабатываемого приложения отличается простотой и содержит лишь необходимые для предоставления функционала решения [29].

Определим, какими функциями должен обладать интерфейс АИС. Система должна отображать табличную информацию со всеми возможностями работы СУБД, такими как полнотекстовый поиск, сортировка по полям, а также предоставлять удобные формы для добавления и редактирования информации. Дизайн интерфейса должен быть минимальным, для простоты использования и повышения скорости работы АИС. На рисунке 14 можно увидеть пример возможного интерфейса АИС. На рисунке видно основное меню, предоставляющее, доступ к редактированию базы данных, доступ к заявкам и исполнениям заявок. Также доступна инструкция и есть возможность

обратиться к разработчикам, или администратору АИС. При переходе между этими пунктами меню (расположено сверху слева), над таблицей и слева снизу появляются дополнительные функциональные кнопки. В данном случае верхняя панель перемещает пользователя между таблицами БД, а нижняя осуществляет поиск, сортировку и вывод на печать. Также, для исключения ошибок ввода и редактирования необходимо предусмотреть ещё две соответствующие кнопки, открывающие формы ввода и редактирования. На рисунке 15 представлен скриншот входа пользователя в систему. После выбора категории, к которой относится сотрудник и ввода пароля ему будут предоставлены соответствующие функции по редактированию, добавлению информации и доступу к основным функциям.

На рисунке 16 представлена форма регистрации заявки. После заполнения и отправки формы соответствующая информация попадёт в БД и станет доступна всем пользователям базы.

Разделы «Заявки на ремонт» и «Выполненные ремонты» доступны только для чтения. Они предоставляют информацию о заявках по различным категориям. Доступны функции вывода отчётов по различным периодам, устройствам, поломкам, отделам, ответственным лицам. Можно подсчитать, сколько и каких запасных частей было использовано за определённый период времени, определить частоту поломок того, или иного устройства, спрогнозировать предстоящие ремонты.

Кроме разграничения прав доступа логинный вход в АИС предоставляет возможность работы с базой данных по принципу «Login-Logout», поддерживающему постоянное соединение с базой данных.

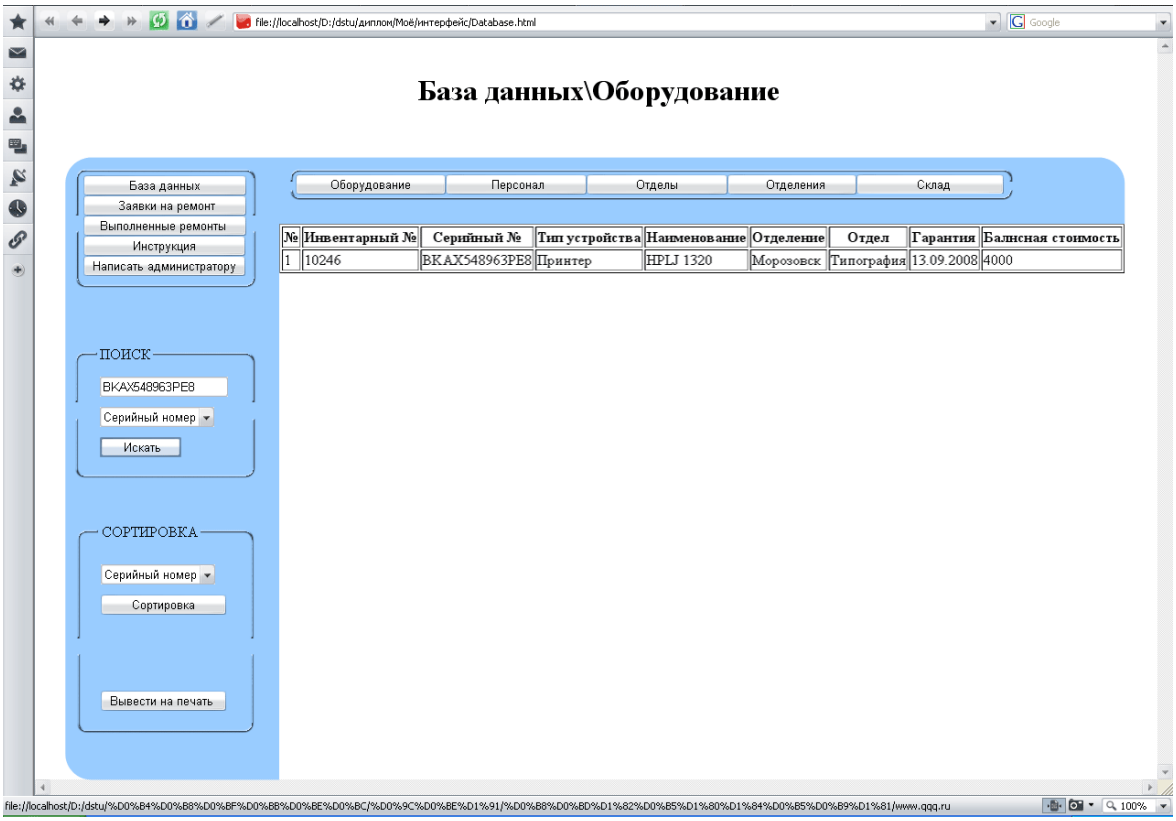

Рисунок 14 – Редактирование БД АИС УИЗРО

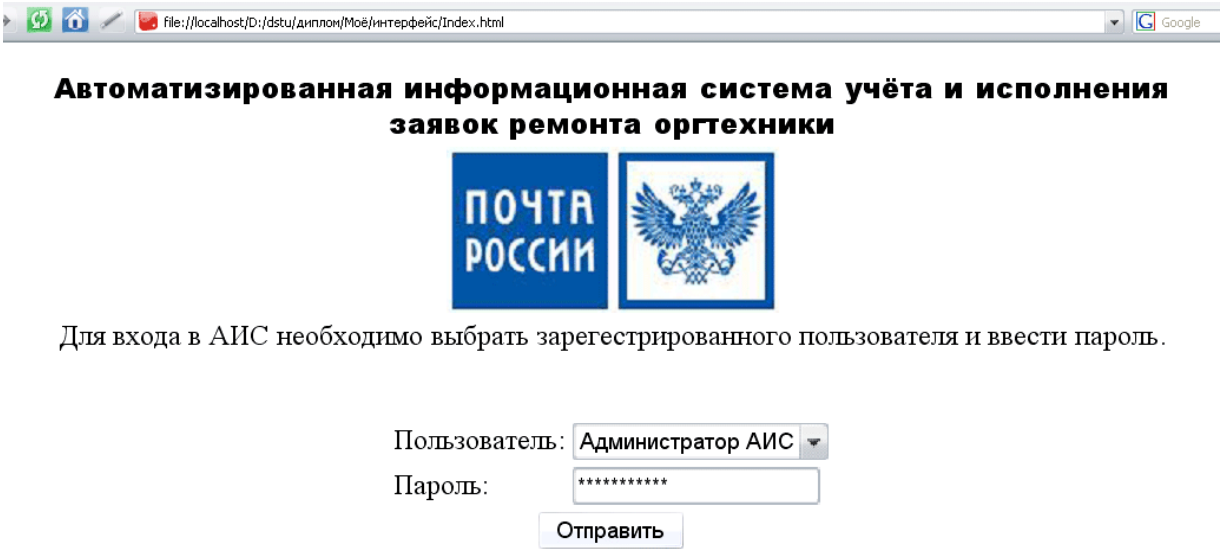

Рисунок 15 – Вход пользователя в АИС УИЗРО

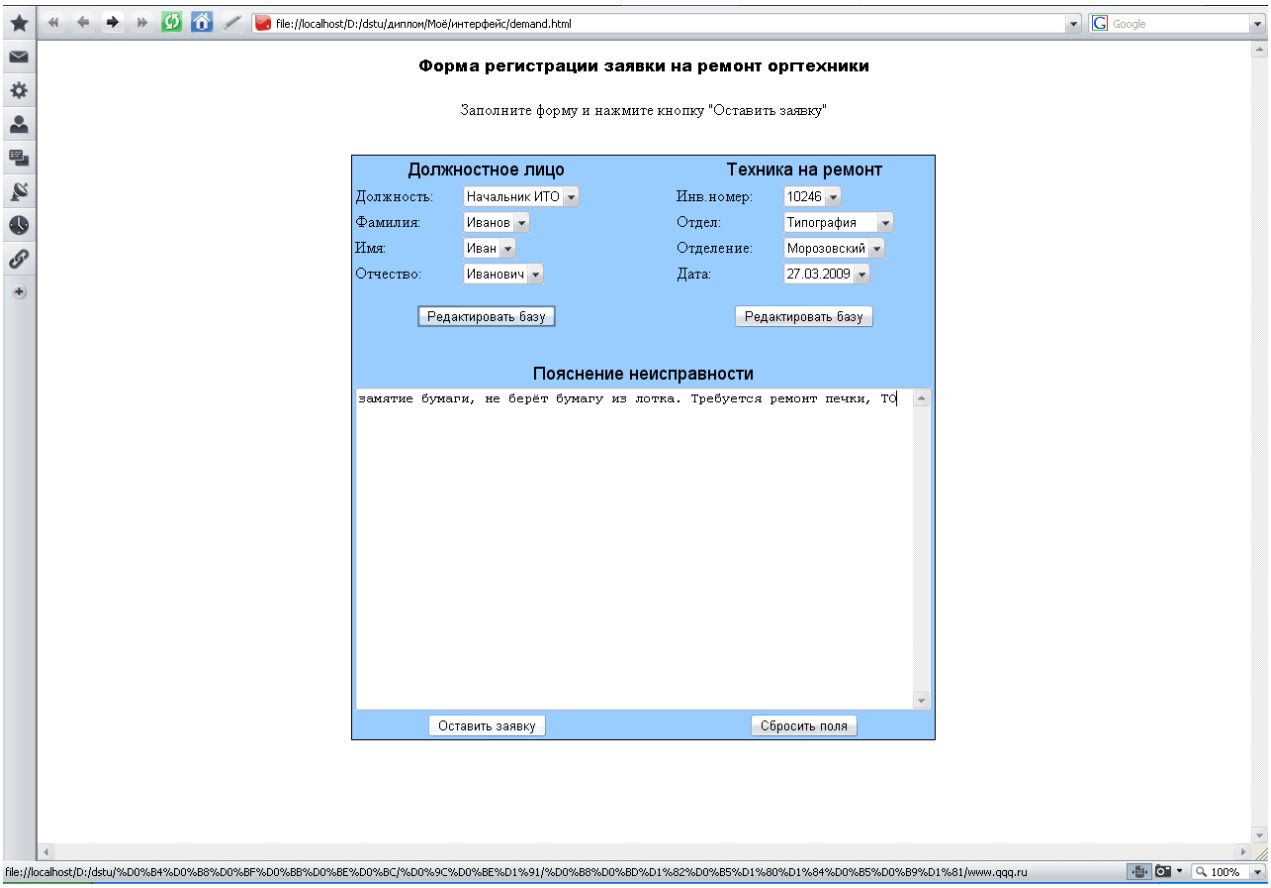

Рисунок 16 – Форма регистрации заявки на ремонт оргтехники

5 Безопасность и экологичность проекта

5.1 Анализ опасных и вредных производственных факторов и мероприятия по обеспечению безопасных и безвредных условий труда

5.1.1 Рабочее место оператора.

Охрана труда – система сохранения жизни и здоровья работников в процессе выполнения трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия (в соответствии со статьей 209 Трудового кодекса Российской Федерации № 197-ФЗ от 30 декабря 2001 года).

Устройство удобного и безопасного рабочего места, отвечающего условиям труда работника, является важным аспектом охраны труда. Основные требования к рабочему месту оператора сформулированы в ГОСТ Р 50923-95 «Дисплеи. Рабочее место оператора. Общие эргономические требования к производственной среде. Методы измерения» от 1 июля 1997 года (переиздан в январе 2008 года).

С учетом данных требований было спроектировано рабочее место оператор для работы в автоматизированной системе, обеспечивающей анализ учета и исполнения заявок ремонта организационной техники.

Данное место обладает следующими характеристиками.

Конструкция рабочего стола обеспечивает удобное размещение на рабочей поверхности необходимого комплекта оборудования и документов. Высота рабочей поверхности стола равна 725 мм. Размеры рабочей поверхности стола: глубина - 650 мм, ширина - 1400 мм. Стол имеет пространство для ног 700 мм, шириной - 600 мм, глубиной на уровне колен - 500 мм и на уровне вытянутых ног - 700 мм. Рабочая поверхность стола не имеет острых углов и краев. Покрытие рабочей поверхности стола выполнено из диффузно отражающего материала.

Рабочий стул подъемно-поворотный и регулируемый по высоте и углам наклона сиденья и спинки, а также расстоянию спинки от переднего края сиденья. Поверхность сиденья имеет ширину и глубину 400 мм. Опорная поверхность спинки стула имеет высоту 300 мм, ширину 380 мм и радиус кривизны в горизонтальной плоскости 400 мм. Расстояние спинки от переднего края сиденья регулируется в пределах от 260 до 400 мм.

Дисплей располагается на столе так, чтобы для рассмотрения изображения в любой части экрана не требовалось изменение положения головы. Дисплей ниже уровня глаз оператора. Угол наблюдения экрана оператором относительно горизонтальной линии взгляда не превышает 60°, как изображено на рисунке 17.

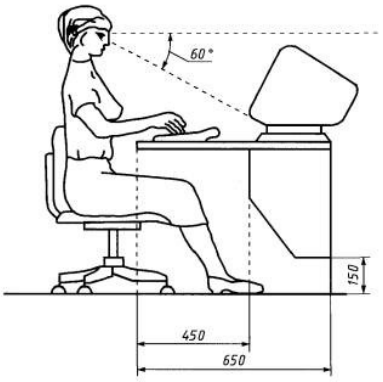

Рисунок 17 – Размещение дисплея на поверхности стола

Освещенность рабочего места оператора на рабочем столе в горизонтальной плоскости от общего искусственного освещения составляет 400 лк. Для освещения зоны расположения документов установлены светильники местного освещения.

Уровень шума не превышает нормативный.

Температура воздуха на рабочем месте в не превышает диапазон в 22 – 25 °С. Относительная влажность воздуха на рабочем месте оператора в среднем составляет 55%.

Скорость движения воздуха около 0,1 м/с.

5.1.2 Оценка напряженности труда.

Напряженность трудового процесса оценивается в соответствии с «Гигиеническими критериями оценки условий труда по показателям вредности и опасности факторов производственной среды, тяжести и напряженности трудового процесса».

На основе комплекса производственных факторов, создающих предпосылки для возникновения неблагоприятных нервно-эмоциональных состояний (перенапряжения), была произведена оценка напряженности труда оператора в процессе трудовой деятельности.

По полученным показателям напряженности трудового процесса были определены классы условий труда, приведенные в таблице 5.1.1.

Таблица 5.1.1 - Классы условий труда оператора по показателям напряженности трудового процесса

№	Показатель напряженности трудового процесса	Характеристика нагрузок оператора	Класс условий труда
Интеллектуальные нагрузки:			
1.	Содержание работы	Решение простых задач по инструкции	2 – допустимый, напряженность труда средней степени
2.	Восприятие сигналов (информации) и их оценка	Восприятие сигналов с последующей коррекцией действий и операций	2 – допустимый, напряженность труда средней степени
3.	Распределение функций по степени сложности задания	Обработка, выполнение задания и его проверка	2 – допустимый, напряженность труда средней степени
4.	Характер выполняемой работы	Работа по установленному графику с возможной его коррекцией по ходу деятельности	2 – допустимый, напряженность труда средней степени

Продолжение таблицы 5.1.1

Сенсорные нагрузки			
5.	Длительность	До 25	1 – оптимальный,

	сосредоточенного наблюдения (% времени смены)		напряженность труда легкой степени
6.	Плотность сигналов (световых, звуковых) и сообщений в среднем за 1 час работы	До 75	1 – оптимальный, напряженность труда легкой степени
7.	Число объектов одновременного наблюдения	До 5	1 – оптимальный, напряженность труда легкой степени
Эмоциональные нагрузки			
8.	Степень ответственности за результат собственной деятельности. Значимость ошибки	Несет ответственность за выполнение отдельных элементов заданий. Влечет за собой дополнительные усилия в работе со стороны работника	1 – оптимальный, напряженность труда легкой степени
9.	Степень риска для собственной жизни и безопасности других лиц	Исключена	1 – оптимальный, напряженность труда легкой степени
Монотонность нагрузок			
10.	Число элементов (приемов), необходимых для реализации задания или в многократно повторяющихся операциях	9-6	2 – допустимый, напряженность труда средней степени
11.	Продолжительность (в сек) выполнения простых заданий или повторяющихся операций	100-25	2 – допустимый, напряженность труда средней степени
12.	Время активных действий (в % к продолжительности смены). В остальное время - наблюдение за ходом производственного процесса	20 и более	1 – оптимальный, напряженность труда легкой степени

Окончание таблицы 5.1.1

13.	Монотонность производственной обстановки (время пассивного наблюдения за	Менее 75	1 – оптимальный, напряженность труда легкой степени

	ходом техпроцесса в % от времени смены)		
Режим работы			
14.	Фактическая продолжительность рабочего дня	8-9 ч.	2 – допустимый, напряженность труда средней степени
15.	Сменность работы	Односменная работа (без ночной смены)	1 – оптимальный, напряженность труда легкой степени
16.	Наличие регламентированных перерывов и их продолжительность	Перерывы регламентированы, достаточной продолжительности: 7 % и более рабочего времени	1 – оптимальный, напряженность труда легкой степени

5.1.3 Расчет СОИ

Для реализации работы автоматизированной системы, обеспечивающей анализ учета и исполнения заявок ремонта организационной техники, применяется визуальное средство отображения информации (далее – СОИ). А именно СОИ на электронно-лучевых трубках – монитор рабочего компьютера.

На основании ГОСТ Р 50948-2001 «Средства отображения информации индивидуального пользования. Общие эргономические требования и требования безопасности» проведен расчет основных светотехнических характеристик СОИ. В результате были вычислены показатели яркости изображения (B_0), контраста ($K_{ПР}$) и число пороговых контрастов (Q).

Необходимые условия: $B_{O.TP} > 10$ кд/м^2; $0{,}65 < K_{ПР.\,TP} < 0{,}95$; $Q_{TP} > 10$.

Так как объект различения темнее фона, то контраст считаем прямым и определяем яркость объекта B_0, кд/м^2, по формуле 5.1.

$$B_0 = \rho_{ип} * \frac{E_н}{\pi}, \quad (5.1)$$

где $\rho_{ип}$ - коэффициент отражения от изображения в информационном поле, определяется согласно ГОСТ 50923-95 «Дисплеи. Рабочее место оператора. Общие эргономические требования к производственной среде. Методы измерения»;

$E_н$ - освещенность поверхности, лк.

При значениях $E_н$ = 400 лк, $\rho_{ип}$ = 0,63, получаем:

$$B_0 = 0{,}63 * \frac{400}{3{,}14} = 80 \text{ кд/м}^2.$$

Фактический прямой контраст $K_{ПР}$ определяем по формуле 5.2.

$$K_{ПР} = \frac{B_ф - B_o}{B_ф}, \quad (5.2)$$

где $B_ф$ – яркость фона, равна 320 кд/м^2.

$$K_{ПР} = \frac{320 - 80}{320} = 0{,}75 \text{ кд/м}^2.$$

Число пороговых контрастов Q определяем по формуле 5.3:

$$Q = \frac{K_{пр}}{K_П} \quad (5.3)$$

где $K_П$ - пороговый контраст, который определяют по номограмме зависимости порогового контраста от яркости фона и углового размера объекта.

Угловой размер заданного объекта α = 5', по номограмме $K_П$ = 0,06 кд/м^2.

Тогда:

$$Q = \frac{0{,}75}{0{,}06} = 12{,}5$$

Так как требуемые условия $B_{О.ТР}$ > 10 кд/м^2; 0,65< $K_{ПР.\ ТР}$ < 0,95; $Q_{ТР}$ >10, то при полученных значениях $B_О$ = 80 кд/м^2, $K_{ПР}$ = 0,75, Q = 12,5 и условия выполняются, то рабочее место считаем прошедшим инженерно-психологическую оценку по основным светотехническим характеристикам.

5.2 Экологичность проекта

Основную опасность для окружающей среды в данном проекте имеет средство реализации работы разрабатываемой автоматизированной системы – персональные компьютеры. И если во время их эксплуатации большинство компонентов, входящих в состав техники, не воздействуют на природу, то при неправильной утилизации они могут загрязнить почву, сточные воды, и даже атмосферу.

Это связано с тем, что в состав техники входят как органические составляющие (пластик различных видов, материалы на основе поливинилхлорида, фенолформальдегида), так и многие металлы. Такие, как свинец, сурьма, ртуть, кадмий, мышьяк, входящие в состав электронных компонентов. При неправильной утилизации переходят под воздействием внешних условий в органические и растворимые соединения и становятся сильнейшими ядами.

Комплексная система утилизации оргтехники позволяет свести к минимуму не перерабатываемые отходы, а основные материалы (пластмассы, цветные и черные металлы) и ценные компоненты (редкие металлы, люминофор, ферриты и др.) вернуть в производство.

В Ростовской области так же есть компании, специализирующиеся на приеме компьютеров для утилизации и переработки. Например:

1) «РегионЭкоХолдинг» , отдел расположен по адресу: г. Ростов-на-Дону, ул. Королева, 3, корп. 1;
2) «Техника+», отдел расположен по адресу: г. Ростов-на-Дону, ул. Текучева, 87 и другие.

5.3 Устойчивость в ЧС

Для создания условий реализации разрабатываемой автоматизированной системы использовались компьютеры.

Их пожарная и взрывная опасность обуславливается работой от электрических сетей. При проектировании рабочего места оператора были

соблюдены необходимые Правила устройства электроустановок (ПУЭ), утвержденные приказом № 204 Минэнерго России от 08.07.2002.

Поэтому при соблюдении всех требований по эксплуатации электрооборудования, практически исключена опасность пожара или взрыва, спровоцированного используемыми аппаратами.

Для исключения возможности возникновения пожара помещение оборудовано извещателем пожарным дымовым «ИП 212-3СУ» и обязательно укомплектовано порошковым огнетушителем.

Так как приборы пожаро- и взрывобезопасны, то не требуется никаких дополнительных мер по изменению строительных конструкции или отделочных материалов.

Средства индивидуальной или коллективной защиты для работы с данным оборудованием не требуются.

В результате можем сделать следующий вывод по разделу: при соблюдении сформулированных требований – как эргономических, так и норм пожарной безопасности – опасность для жизни и здоровья людей сводится к минимуму. При осуществлении правильной утилизации и переработки перечисленных компонентов воздействие данного проекта на окружающую среду также минимизировано.

Таким образом, можем считать проект безопасным с точки зрения экологичности и безопасности жизнедеятельности.

ЗАКЛЮЧЕНИЕ

В заключение отмечу, что итогом работы стала не только описанная выше АИС, но и средства для создания класса клиент-серверных учетных систем с web-интерфейсом. Был использован подход, при котором данные, пользовательский интерфейс и управляющая логика разделены на отдельные компоненты. Предлагаемые современные средств разработки позволили автоматически создавать приложения по принципу «Модель-Вид-Контроллер». Построенные по технологии ORM они позволяют связать реляционную базу данных с концепциями объектно-ориентированных языков программирования, создавая виртуальную объектную БД. Применение фреймворка облегчает и ускоряет разработку проекта, предоставляя все современные достижения программирования.

Применение Веб-технологий позволяет создать приложение, предоставляющее удалённый доступ к системе. Благодаря применению в качестве клиентского ПО web-браузера и переноса обработки информации на сервер достигается необходимый уровень надежности защиты информации и простоты обслуживания системы. Инструментарий позволяет строить приложения, выполняющие необходимые операции с данными, как базовые (ввод, чтение, редактирование, удаление), так и сложные (отображение согласно семантике, построение отчетов).

При разработке АИС, так же были использованы средства функционального моделирования, такие, как AllFusion ERwin Data Modeler и AllFusion ERwin Process Modeler. Построенные функциональные диаграммы помогли определить структуру автоматизируемого процесса с точки зрения того, какой он должен быть и выявить места требующие использования автоматизированных информационных систем. Проектирование модели данных помогает решить вопрос со структурой базы данных и упростить процесс её создания.

Исследование рынка в области учётных и автоматизированных информационных систем показали отсутствие подобных решений. Внедрение и кастомизация более мощных автоматизированных систем требует больших финансовых затрат и времени, чем внедрение АИС УИЗРО. Это говорит о конкурентоспособности предлагаемой АИС. При всех своих достоинствах АИС является относительно не дорогим решением.

Так же был произведен анализ опасных и вредных факторов, сопряжённых с использованием АИС УИЗРО. Расчёты не выявили никаких превышений норм.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. http://www.iteam.ru/publications/it/section_54/article_2726/ - Комплекс полноценности.
2. http://www.osp.ru/nets/1997/06/142618/#part_1 - Многоуровневые системы клиент-сервер.
3. http://www.osp.ru/pcworld/2005/02/169733/ - «Тонкий клиент» Грязнов.
4. http://belani.narod.ru/1/Lklser2.htm - Архитектура клиент-сервер: определение, предпосылки для применения, плюсы и минусы.
5. http://www.mstu.edu.ru/education/materials/zelenkov/ch_7_1.html - Архитектура "клиент-сервер".
6. http://www.citforum.ru/seminars/cis99/useweb.shtml - Применение Web-технологий в информационных системах.
7. http://magazine.keyintegrity.com/2006/11/preimushhestva-soa-dlya-biznesa/ - Преимущества SOA для бизнеса. Владимирова Ольга.
8. http://www.itpedia.ru/index.php/SOA - SOA.
9. http://www.bytemag.ru/articles/detail.php?ID=12164 - SOA: переход от теории к практике. Андрей Колесов.
10. http://www.softholm.com/news/software/article_64.html - Автоматизированная система «Учет оргтехники и расходных материалов».
11. http://www.ccc.ru/magazine/depot/98_02/read.html?web2.htm - Архитектура клиент–сервер или Web: выбор разработчика.
12. http://ru.wikipedia.org/wiki/Шаблоны_проектирования - Шаблоны проектирования.
13. http://chtivo.webhost.ru/articles/mvc.php - MVC для начинающих.
14. http://rsdn.ru/article/patterns/generic-mvc.xml - Обобщенный Model-View-Controller.

15. http://ru.wikipedia.org/wiki/PostgreSQL - PostgreSQL. Материал из Википедии.

16. http://postgresmen.ru/ - поддержка PostgreSQL в России.

17. http://www.citforum.ru/database/osbd/contents.shtml - Основы современных баз данных. С.Д. Кузнецов, информационно-аналитические материалы Центра Информационных Технологий.

18. http://ru.wikipedia.org/wiki/ORM - ORM. Материал из Википедии.

19. http://ru.wikipedia.org/wiki/Фреймворк - Фреймворк. Материал из Википедии.

20. http://ru.wikipedia.org/wiki/Django - Django. Материал из Википедии.

21. http://djbook.ru/ - Введение в Django.

22. http://ru.wikipedia.org/wiki/Apache - Apache. Материал из Википедии.

23. http://apachedev.ru/2006/03/12/the-apache-modeling-project-vvedenie/ - Архитектура Apache. Проект «The Apache Modeling Project».

24. http://apache.tversu.ru/ - Проект по переводу документации к Apache на русский.

25. www.ca.com/ru/ - Computer Associates. Россия.

26. http://www.interface.ru/home.asp?artId=2510 - Функциональное моделирование с использованием продукта AllFusion Process Modeler 4.1.4.

27. http://www.interface.ru/home.asp?artId=101 - AllFusion ERwin Data Modeler 7 (ERwin). Проектирование, документирование и сопровождение баз данных и хранилищ данных.

28. http://www.info-system.ru/designing/methodology/erwin/erwin_design_db.html - Проектирование баз данных с ERwin. Зайцев С.Л., к.ф.-м.н.

29. http://www.inteltec.ru/publish/articles/objtech/4kx6_9.shtml - WEB-интерфейс объектной СУБД.