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

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ ДЛЯ ЭЛЕКТРОННОГО РЫНКА ИННОВАЦИЙ НИУ БелГУ

Выпускная квалификационная работа
обучающегося по направлению подготовки
09.03.02 Информационные системы и технологии
Очной формы обучения, группы 12001509
Тюрина Кирилла Владимировича

Научный руководитель
к.э.н., ст. преподаватель
Нестерова Е.В.

БЕЛГОРОД 2019

РЕФЕРАТ

Разработка информационной системы для электронного рынка инноваций НИУ БелГУ. – Тюрина Кирилла Владимировича, выпуская квалификационная работа Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 63, включая приложение 77, количество рисунков 23, количество таблиц 4, количество формул 7, количество использованных источников 47.

КЛЮЧЕВЫЕ СЛОВА: информационная система, база данных, НИОКР, рынок инноваций.

ОБЪЕКТ ИССЛЕДОВАНИЯ: учет заявок НИОКР электронного рынока инноваций НИУ «БелГУ».

ПРЕДМЕТ ИССЛЕДОВАНИЯ: разработка информационной системы для электронного рынка инноваций НИУ БелГУ.

ЦЕЛЬ РАБОТЫ: усовершенствование системы рынка инновации НИУ БелГУ за счет разработки информационной системы.

ЗАДАЧИ ИССЛЕДОВАНИЯ: рассмотреть основные теоретические аспекты НИОКР; проанализировать рынок инноваций НИУ БелГУ и разработать бизнес-модели работы с заказчиками как будет; разработать функциональные требования к информационной системе для электронного рынка инноваций НИУ БелГУ; разработать информационную систему для электронного рынка инноваций НИУ БелГУ.

МЕТОДЫ ИССЛЕДОВАНИЯ: эмпирические методы: изучение разнообразных источников информации, анализ полученных сведений; теоретические методы: анализ, синтез, классификация.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы была спроектирована и реализована информационная система для электронного рынка инноваций НИУ БелГУ.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

НИОКР – один из факторов, обусловливающих экономический прогресс в стране. Научно-исследовательская работа в России сегодня испытывает трудные времена потому, что до начала реформ 90-х годов, значительные инновации осуществлялись государством. По этой причине, специфика сформировавшейся в настоящий период ситуации состоит в том, что в государстве есть крупные научно-технические заделы, оригинальная научно-производственная база и высококвалифицированные кадры, но из-за всемирного экономического кризиса имеется крайне слабая ориентация этого инновационного потенциала на реализацию научных достижений. Важность государственного регулирования НИОКР обусловлено их увеличивающимся значением как для экономики, так и для людей в целом. Государство обязано регулировать научный сектор, так как в настоящее время он в высокой степени определяет перспективы развития государства [47].

Актуальность темы выпускной квалификационной работы (ВКР) заключается в том, что практически каждая организация использует в своей работе сеть Интернет. В результате развития информационных технологий поменялись способы ведения бизнеса.

За последнее время объем и оборот информации в разных сферах деятельности человека – экономической, политической, вырос. А процесс обработки, использования и накопления информации ускоряется год от года. Ученые считают, что через небольшой промежуток времени объем используемой информации увеличится вдвое, поэтому возникает необходимость использования автоматизированных средств обработки, хранения и распределения данных и информации.

Работа финансовой службы крупных организаций напрямую зависит от уровня оснащенности информационными средствами на базе компьютерных систем конкретных организаций.

Необходимо понимать, что компьютерное ведение дел отличается от обычного, так как позволяет облегчить работу, уменьшая время на оформление и создание документов, и обработку всей информации, которая относится к торговой деятельности. Конечно, все отчеты можно сделать и без помощи компьютера, но на их подготовку уйдет много времени, и они уже станут не нужны. Или же нужно будет увеличить количество сотрудников, это приведет к увеличению расходов на зарплату и к нулевой прибыли для самой организации. Следовательно, применяя компьютер, мы увеличиваем скорость расчетов и улучшаем качество готовых отчетов. Это дает возможность улучшить схему построения бизнеса.

В настоящее время практически все организации работают в условиях сложного финансово-экономического кризиса. И для дальнейшей успешной работы необходимо правильно организовать внутреннюю деятельность организации на всех уровнях, так как сейчас информационные технологии могут предоставить такие возможности. Из вышесказанного, становится ясно, что автоматизация деятельности является одной из основных задач для руководителей организаций.

Возникают следующие трудности, что актуализует исследование:

- сроки исполнения проектов НИОКР (заказов);
- производительность труда;
- себестоимость работ и продукции;
- нет единой информационной среды.

Таким образом, объект исследования ВКР - учет заявок НИОКР электронного рынка инноваций НИУ «БелГУ», предмет исследования – автоматизация учета заявок НИОКР НИУ БелГУ.

Цель ВКР - усовершенствование системы рынка инновации НИУ БелГУ за счет разработки информационной системы.

Для достижения поставленной цели необходимо решить следующие задачи:

- рассмотреть основные теоретические аспекты НИОКР;

- проанализировать рынок инноваций НИУ БелГУ и разработать бизнес-модели работы с заказчиками как будет;

- разработать функциональные требования к информационной системе для электронного рынка инноваций НИУ БелГУ;

- разработать информационную систему для электронного рынка инноваций НИУ БелГУ.

ВКР состоит из 3 разделов, 4 таблиц, 7 формул и 23 рисунков.

1 Теоретическая часть

1.1 Характеристика проекта НИОКР

Новшества помогают государству получать большую прибыль, используя новые технологии. Развитые страны активно поддерживают развитие ноу-хау: привлекают учёных, помогают развиваться малому и среднему бизнесу, делают налоговые льготы на ведение деятельности. Объем новых технологий, которые поступают из России на мировой глобальный рынок, малозначителен. Внутри нашей Родины он также малозаметен – все инновации привезены из-за границы и адаптированы для российских граждан.

Рынок новых технологий в России формируется исходя из тех обстоятельств, когда предприниматели не способны самостоятельно создавать новшества, а пытаются получить их из прошлых результатов научно-исследовательских работ времен СССР. Из-за этого рынок данной сферы формируется специфическим образом.

Управление научно исследовательскими и опытно-конструктивными работами (НИОКР) - это выделение решений в постоянно изменчивых рыночных условиях, постоянное описание программы НИОКР, изменение ее в целом или некоторых частей. Возможны некие технические проблемы, потребность в разделении ресурсов, обновление оценки возможностей.

Любой проект начинается с постановки цели. Т.к. итоговый успех показывается на рынке, то и цели должны быть выражены с этой потребностью. Изначально, это рыночный сегмент и его совокупные параметров (размер, стоимость, требования по эффективности и длительности вывода продукта). Продукт сам по себе обязан быть значимым в рамках эффективности, цене и периоду выхода.

Система контроля проекта обязана быть адекватной его объёму, проблематичности, уровню неопределённости, месту расположения в НИОКР. Также она должна поддерживать:

- уровень прогресса в решении любой задачи, расходы и периоды работ;

- определение тех задач, реализация которых выходит из графика, оценку причин этого для совокупной работы в проекте;

- контроль развития проекта в целом относительно уровня затрат и даты окончания.

Сложностью контроля НИОКР становится эффективное разделение ресурсов. Это поясняется рядом причин. Важно, чтобы совокупная величина ресурсов в сфере НИОКР стала стабильной по ходу времени. Ресурсы переводятся в любое оборудование, которые имеет четкую цену вне зависимости от того, работает оно или нет, либо в счет оплаты труда; и то и другое – ресурсы заменяемые и особые. Каждый проект подразумевает свою комбинацию подобных ресурсов, причем из-за уровня непонятности проекта заранее их разделить не представляется возможным.

Стоимостной критерий проекта тоже нельзя определить точно. Но важно знать, что его форма так или иначе отражается руководством НИОКР.

Дата ухода продукта – это функция контроля в рамках принципов, включенных на начальных стадиях проекта; длительность ЖЦ товара зависит от срока его прихода и ухода с рынка. Поэтому менеджмент НИОКР ориентируется на сокращении периода НИОКР. Четкая дисциплина во времени должна быть принесена еще в начале программы создания проекта.

Применение нового продукта в рамках производства (параметра производства) очень нечасто получается без трудностей. И делят их на пару групп:

- сложности, которые связаны с мощностью производства для нового продукта;

- сложности публикации итогов проекта, связанные с поддержание расходов, гарантирующих приход ожидаемой прибыли.

Задержки из-за сложности покупки нового оборудования, набора и обучения сотрудников, проблем обслуживания коммуникаций оказывают воздействие на стабильность и должны быть учтены в проекте. В рамках оценки, важно определить его параметры, которые смогут вызвать проблемы у производителя.

Финальные издержки производства идут из материалов и комплектующих изделия, используемых технических приемов, капитальных вложений и реализация линии производства. Эти издержки выражены и успешностью продаж.

По итогу, к числу базовых производственных факторов, приносящих удачу для проекта, относят:

- отвечающие требованиях производства технологии;
- положительный баланс мощностей производства;
- обеспеченность рынка материалами и комплектующими к продукции;
- открытость сторонних ресурсов;
- универсальность производства, его способность успешно выпускать новую продукцию с издержками, которые помогут держать цену;
- уровень применяемых технологии и оборудования.

Безусловно, часть критериев оценки проектной деятельности не отнести к научной области.

Процесс оценки, там, где он реализован эффективно, становится частью инновационного процесса, что помогает учесть мнение руководителей компании на стартовых этапах НИОКР, что ведет к переходу проекта от НИОКР к производству и управлению.

Под результативностью можно понимать достижение цели контроля объектом. Параметры эффективности часто объединяют, основываясь на параметрах рыночной, технической и эффективности [7].

Результативность совокупных решений по оценке проектов поддерживается привлечением специалистов, которые имеют некий багаж знаний в некой научно-производственной области.

Применение описанного метода оценки проектов не будет считаться математическим расчетом явной или неявной эффективности его реализации, а поможет выбрать компании форму параметров оценки внедрения разработок. А успешность реализации инновационного проекта идет от достоинств и рыночных условий, от качества контроля работы управления [8].

Уровень продукции становится важным показателем роста эффективности производства, что выражается уровнем становления науки, уровнем качества применении её результатов и этапом подготовки сотрудников.

Для части сфер произведение оценки качества продукции выражено в совокупности параметров [4].

Для производителя продукции: минимизация потерь от брака и рекламаций; рост эффективности получения товара; рост уровня конкуренции; рост прибыли от реализации продукции улучшения ее качества; рост имиджа фирмы.

Для покупателя продукции: удовлетворение спроса минимальным числом изделий отличного качества; минимизация затрат в рамках эксплуатации продукции; улучшение и обновление товарного ассортимента изделий; улучшение условий труда в сфере потребления.

Для страны: рост научно-технического прогресса (НТП) в промышленности; новые экспортные возможности; улучшение потребностей населения; повышения уровня окружающей среды.

Индикатор уровня продукции – количественный параметр свойств продукции, которые отражают ее качество, описывается применительно к некоторым условиям её разработки и эксплуатации.

Параметры качества для этих уровней различны и разделены по однородности показателей.

Уровень итогов производственной деятельности становится важным фактором уровня конкурирования предприятия.

Помимо оценок по описанным критериям важно понять рейтинговые веса как некоторых, так и групп факторов, которые оказывают некое влияние на проектную работу, а затем объединять итоги оценок, к примеру, аддитивным или мультипликативным методами.

Для подтверждения и последующего применения параметров лучше всего создать методический аппарат, основанный на современных методах выделения критериев уровня результативности созданных проектов в менеджменте НИОКР.

Для решения вопроса, нужна ли гибкая (AGILE) методология разработки в данной конкретной команде/компании, необходимо понимать основные требования, выдвигаемые к результату и к используемым ресурсам. Существует ряд случаев, когда попытка применения AGILE методологии может повредить процессу разработки проекта ОКР и итоговому результату:

- если команда/компания не готова финансировать дополнительные ресурсы на коммуникацию. Регулярные встречи в рамках AGILE отнимают время и снижают время, используемое на непосредственную разработку, в том числе если команда использует удаленные средства коммуникации;

- если жестко зафиксированы сроки выполнения конкретного задания. Так как в AGILE методологии разработки измеряется спринтами, то погрешность готовности проекта может измеряться 2-3 неделями;

- AGILE методология про гибкую разработку и менее формализованные технические задания. Если требуется писать подробную документацию, как например, в компаниях, требующих высокоточные системы, применение AGILE не оправданна.

Рассмотрим, как происходит процесс разработки каждого из таких проектов. Все начинается с появления идеи по проекту, которая позже выливается в техническое задание на проект ОКР. Техническое задание композируется на итерационные слои. Например, слой может включать в себя круглый сценарий, но с ограниченной функциональностью. Следующий слой для этого сценария будет включать в себя больше функциональных

возможностей и так далее. Все тестовые сценарии публикуются параллельно с тем, как разрабатывается проект. После появления первой MVP (Minimal Value Product) версии любого из функционалов, его отдают в тестирование.

Пока разработчики переключаются на следующий функционал в рамках обновления, тестировщики заводят баги в специальные системы учета дефектов. Благодаря короткой итерации сделано-протестировано, разработчикам удается не выпадать из контекста и время на исправление дефектов существенно сокращается. По завершению функциональных тестов последней законченной feature в рамках апдейта, тестировщики сообщают о всех найденных багах. Если среди найденных багов нет тех, которые могут блокировать полную проверку продукта, запускается полная проверка всего проекта, включая уже используемый функционал. По итогу полной проверки выявляются дефекты в тех местах, где функционал был изменен из-за новых изменений. Составляется список багов по обновлению и приоритезируется. Все дефекты с высоким уровнем критичности правятся, а дефекты низким уровнем критичности и низким процентом воспроизведения переносятся в Backlog. После всех исправлений тестировщики проверяют исправления и прогоняют легкие регрессионные тесты на основании review проекта, который подвергся изменению.

Каждые день команда встречается на ежедневных 15ти минутных летучках, на которых каждый рассказывает очень коротко о своем прогрессе, есть ли у него какие-либо проблемы. Вся разработка построена в виде 2х недельных итераций к концу каждой итерации команда демонстрируется рабочий продукт с ограниченной функциональностью. По итогу 2х недельных итераций команда так же собирается на ретроспективную встречу, где обсуждаются вопросы, связанные с проблемами в разработке в этот период.

1.2 Анализ работы рынка инноваций НИУ БелГУ

Работа любого объекта – это процесс взаимодействия его элементов во времени и в пространстве, который обеспечивает выполнение поставленных целей в условиях воздействий извне и с учетом имеющихся ресурсов. При анализе работы объекта из-за его сложности разбивают системы на части. Этот процесс называется декомпозицией. Разбивать систему на части можно пока выделенный элемент не перестает выполнять в системе каких-либо функций.

Лучше всего делить систему на основе выполняемых функций, а также существующей линейной структуры управления. Функциональная иерархия предполагает специализацию по отдельным функциям управления на всех уровнях этой иерархии.

Теперь более подробно рассмотрим, как осуществляется работа с заказчиками в НИУ БелГУ на выполнение инновационных тендеров. Схема с использованием методологии IDEF, характеризующая процессы работы с заказчиками НИУ БелГУ, приведена на рисунке 1.1.

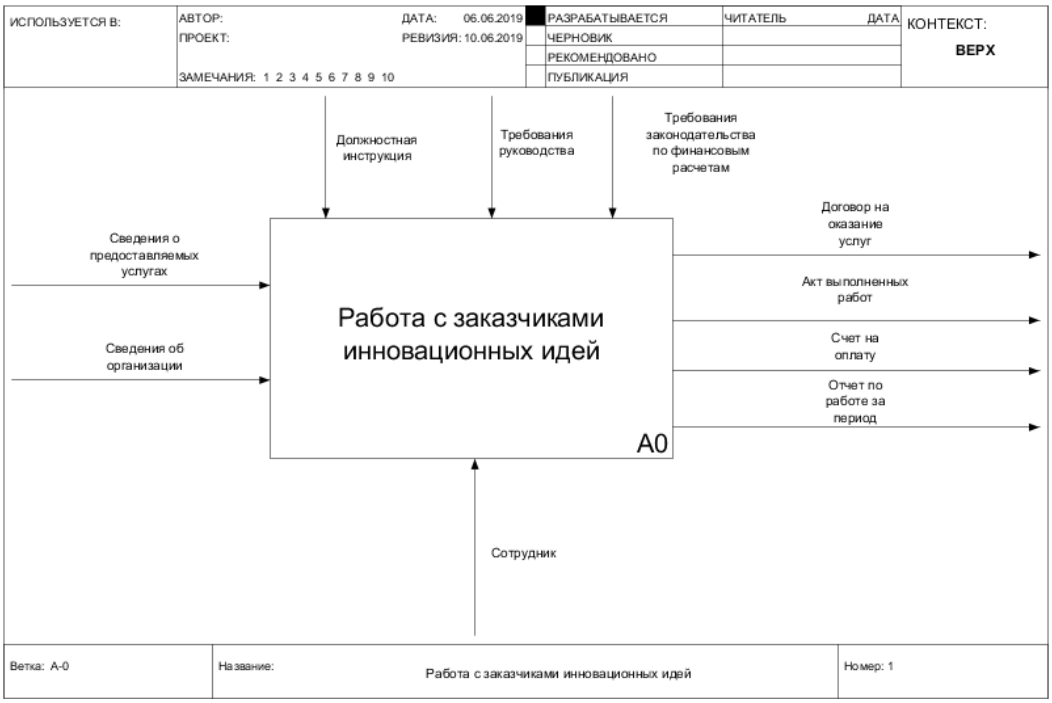

Рисунок 1.1 - Схема процесса работы с заказчиками КАК ЕСТЬ

Декомпозиция процесса работы с заказчиками инновационных идей представлена на рисунке 1.2.

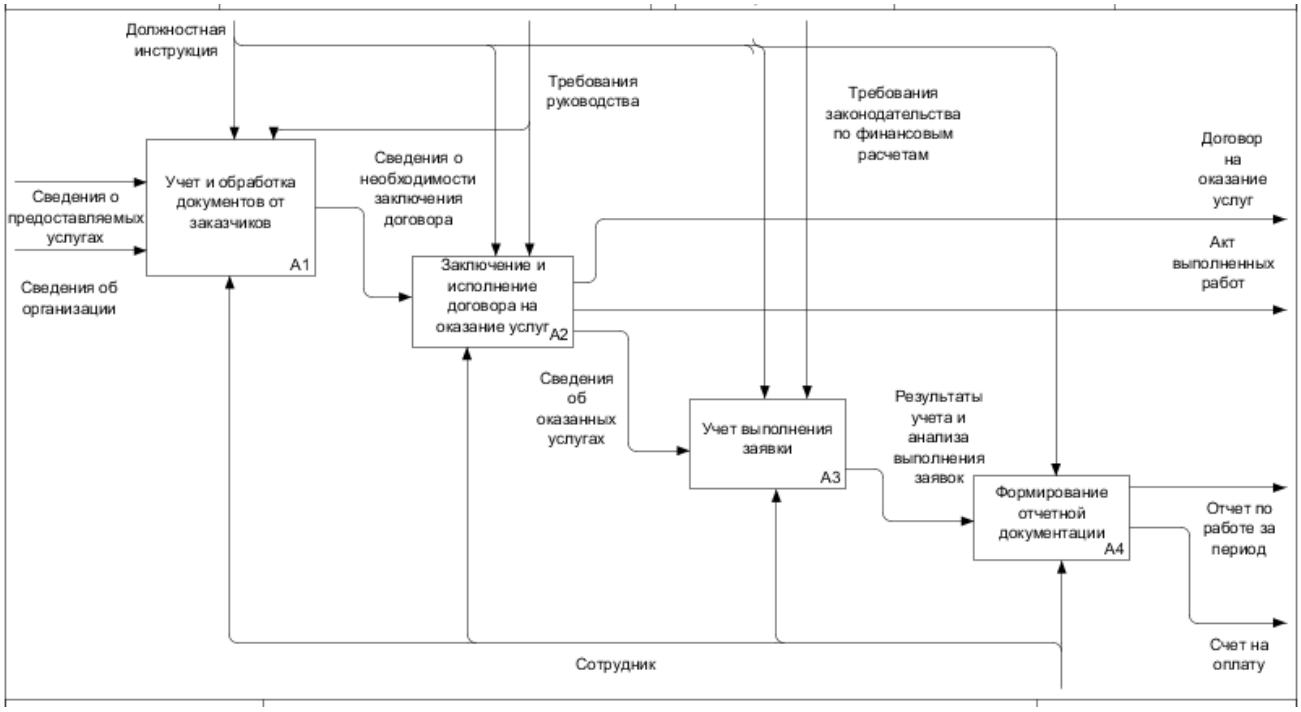

Рисунок 1.2 - Декомпозиции процесса обработки заявок (обращений)

Прежде всего, сотрудники, работающие с заказчиками, отвечают за поддержание постоянной связи с существующими заявителями. В их функции входит:

- формирование и ведение банка данных о заказчиках;

- интерактивный (не прямой) диалог с заказчиков посредством телефонной связи, электропочты, традиционной почты, общения в социальных сетях;

- обеспечение соблюдения интересов заявителей при оказании услуг.

При первичном обращении заявителя сотрудник учитывает полученную информацию, и выясняет, какие услуги он может оказать, на какой тендер претендует, в результате чего формирует список полученных заявок на каждый заказ (тендер).

В настоящее время в НИУ БелГУ учет заявок на тендеры, самих тендеров, а заказ заявок от заявителей на выполнение тендеров ведется исключительно в файле Excel.

Такой способ ведения учета имеет следующие недостатки:

- занесение информации в файл происходит вручную, что снижает скорость обработки информации;

- информация хранится в одном документе, что затрудняет доступ к нему других заинтересованных лиц;

- невозможно проанализировать поступающую информацию по заявкам, к примеру, выделить количество заявок от одного заявителя;

- файл доступен только на одном компьютере и не может быть одновременно быть редактируемым несколькими пользователями;

- существует опасность потери файла со всей информацией.

Для того, чтобы избежать вышеперечисленных недостатков, планируется создать информационную систему в виде веб-приложения, которая бы позволяла пользователю самостоятельно создавать заказы и заявки на тендеры, обрабатывать их.

1.3 Функциональные требования к информационной системе для электронного рынка инноваций

Система для электронного рынка инноваций, называемая далее Системой, содержит Программную, Организационную и Техническую часть.

Информационная система представляет собой полноценное программное обеспечение и разрабатывается собственными силами НИУ БелГУ.

Цель и назначение создания системы:

ИС предназначена для автоматизации управления электронным рынком инноваций:

- осуществление документального учета необходимых для закупа товаров, услуг;

- создание и реализация механизма ценообразования с учетом специфики регионального и местного законодательства;

- разработка электронного документооборота с заказчиками;

- разработка и реализация функций оформления заявки, создания и согласования договора;

- проведение регистрация контрактов и анализа закупочной деятельности;

- организация электронного архива проведенных торгов.

Целями создания системы являются:

- повышение эффективности управления электронным рынком инноваций;

- повышение оперативности и качества работы отдела экономического анализа и прогнозирования;

- улучшение контроля документооборота в отделе.

Характеристика объекта автоматизации:

- рабочие места работников отдела оснащены компьютерами;

- персонал должен иметь навыки работы на ПК;

- в отделе функционирует локальная сеть между компьютерами;

Требования к системе:

Программная часть системы должна представлять собой «Программный Продукт» и иметь следующую архитектуру:

- система должна быть централизованной, т.е. все данные должны располагаться в центральном хранилище и иметь трехуровневую архитектуру.

- смежными системами являются ИС планирования.

Источниками данных для Системы должны быть:

- ИС обеспечения бюджетного процесса (СУБД MS SQL 2012);

- информационно-справочная система - обмен файлами ОС определенного формата;

- информационная система обеспечения бюджетного процесса - интеграция «точка – точка»;

Требования к функциям (задачам) системы:

- централизованное хранение файлов (документов, архивов);

- составление и редактирование заявок на закупки для муниципальных нужд – наименование номенклатуры, объем закупки, цена единицы номенклатуры, стоимость закупки;

- мониторинг выполнения закупок – вычисление параметров прохождения заказов на закупку необходимой номенклатуры – время выполнения заявки, время прохождения этапов, время фиксации в системе заявки;

- просмотр и изменение документов и их архивов, которые осуществляют операторы системы. Архив представляет собой структурированное множество пройденных и утвержденных в системе заявок, предназначенных для длительного хранения и подлежащие специальной операции – архивированию;

- обмен документами и заданиями (заявки, информация о товарах, ценах, сроках поставки, проведения аукционов и утверждения) между структурными подразделениями и операторами и заданиями;

- создание и изменение электронных архивов проведенных торгов. Архив представляет собой цифровое структурированное множество пройденных и утвержденных в системе торгов, предназначенных для длительного хранения и подлежащие архивированию.

1.4 Постановка задачи

На основании анализа работы рынка инноваций, особенностей проектов НИОКР для улучшения эффективности работы с заказами по НИОКР и руководствуясь требованиями, предъявляющим к инновациям НИУ «БелГУ» была поставлена цель: усовершенствование системы рынка инновации «БелГУ» за счет разработки информационной системы

Для реализации цели были сформулированы следующие задач:

- рассмотреть основные теоретические аспекты НИОКР;

- проанализировать рынок инноваций НИУ «БелГУ» и разработать бизнес-модели работы с заказчиками как будет;

- разработать функциональные требования к информационной системе для электронного рынка инноваций НИУ «БелГУ»;

- разработать информационную систему для электронного рынка инноваций НИУ «БелГУ»

Выводы по первому разделу

В первом разделе ВКР проведен анализ предметной области, а именно рассмотрена характеристика проектов НИОКР, построена модель бизнес-процессов работы, с заказчиками, которые проходят без ИС.

2 Разработка проекта информационной системы

2.1 Разработка модели бизнес-процессов работы с заказчиками «Как будет»

В первом разделе на модели «Как есть» были выявлены недостатки, поэтому было решено автоматизировать работу с заказчиками.

Для разработки системы была составлена информационная модель, которая показа на рисунке 2.1 в нотации IDEF0, характеризующая процесс работы с заказчиками после внедрения информационной системы (ИС).

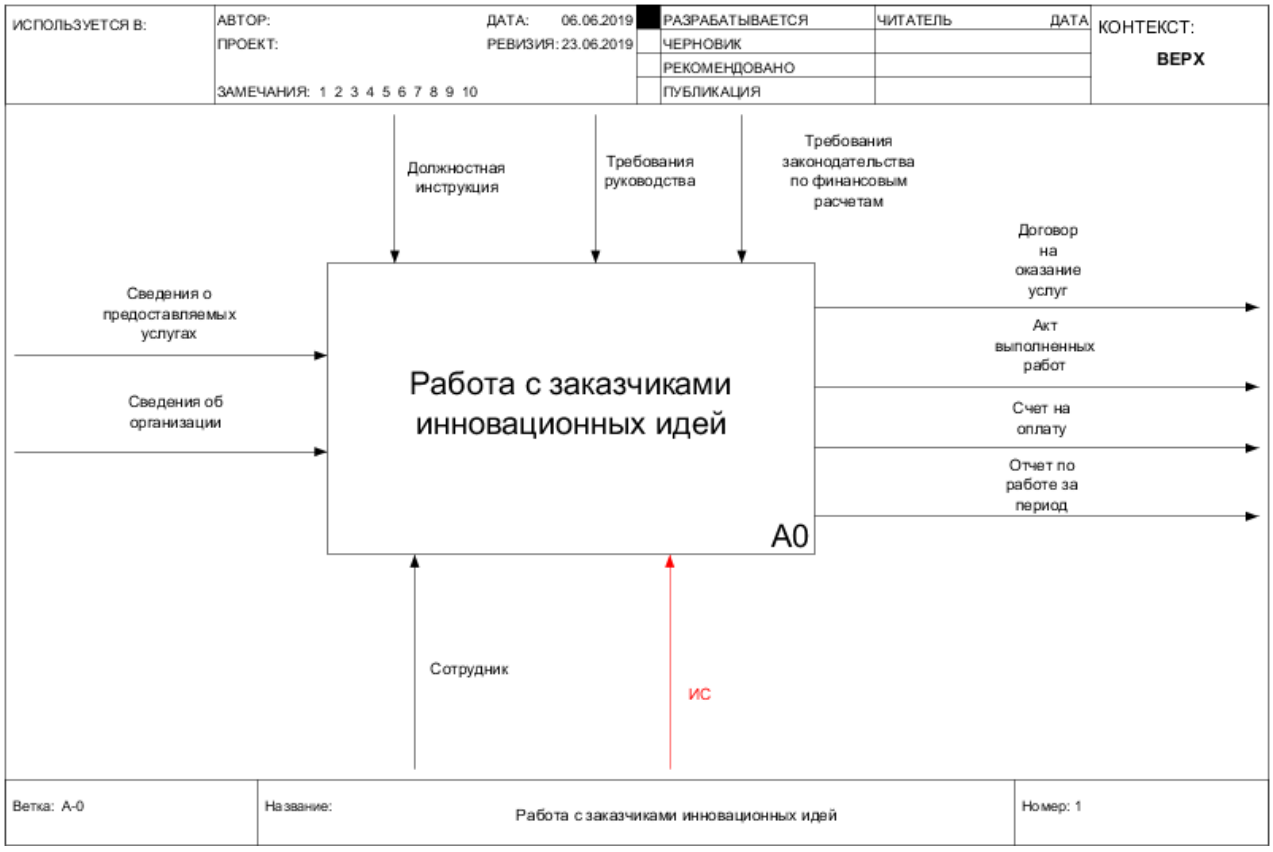

Рисунок 2.1 - Характеристика процесса работы с заказчиками после внедрения информационной системы

Таким образом, в данном случае (по сравнению с функциональной моделью «Как есть», представленной на рисунке выше), добавляется новый механизм — ИС.

На рисунке 2.2 представлена схема декомпозиции процесса учета работы с заказчиками после внедрения ИС.

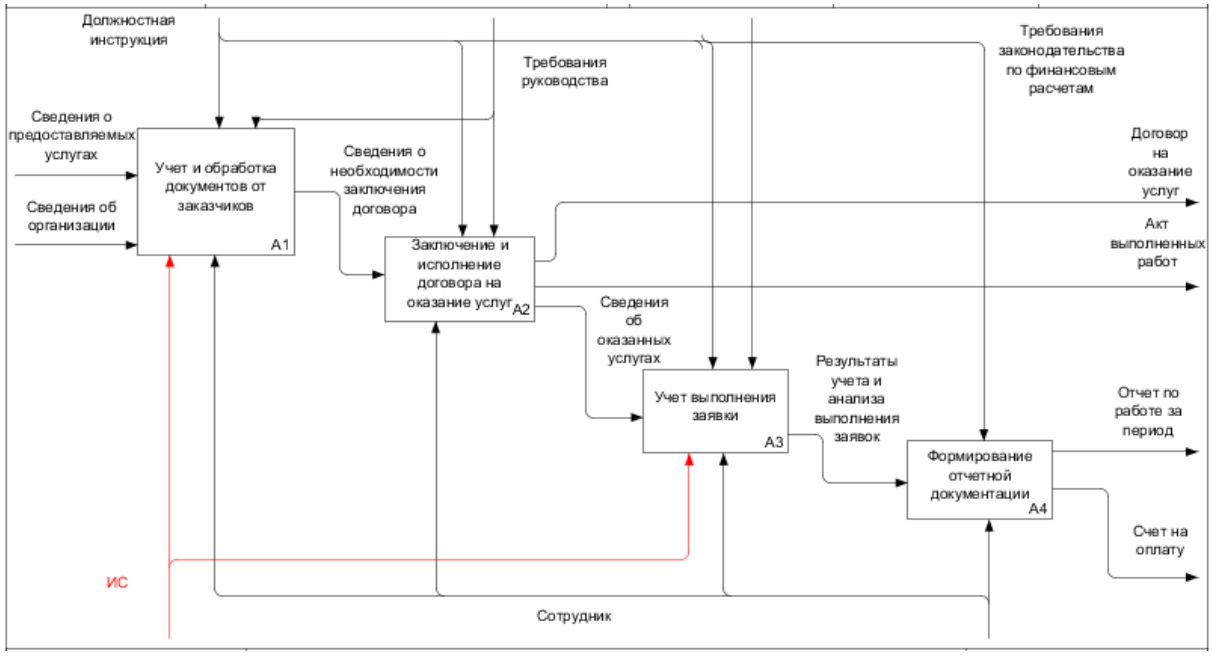

Рисунок 2.2 - Декомпозиции процесса работы с заказчиками «Как будет»

Схема декомпозиции процесса ввода документов в систему представлена на рисунке 2.3.

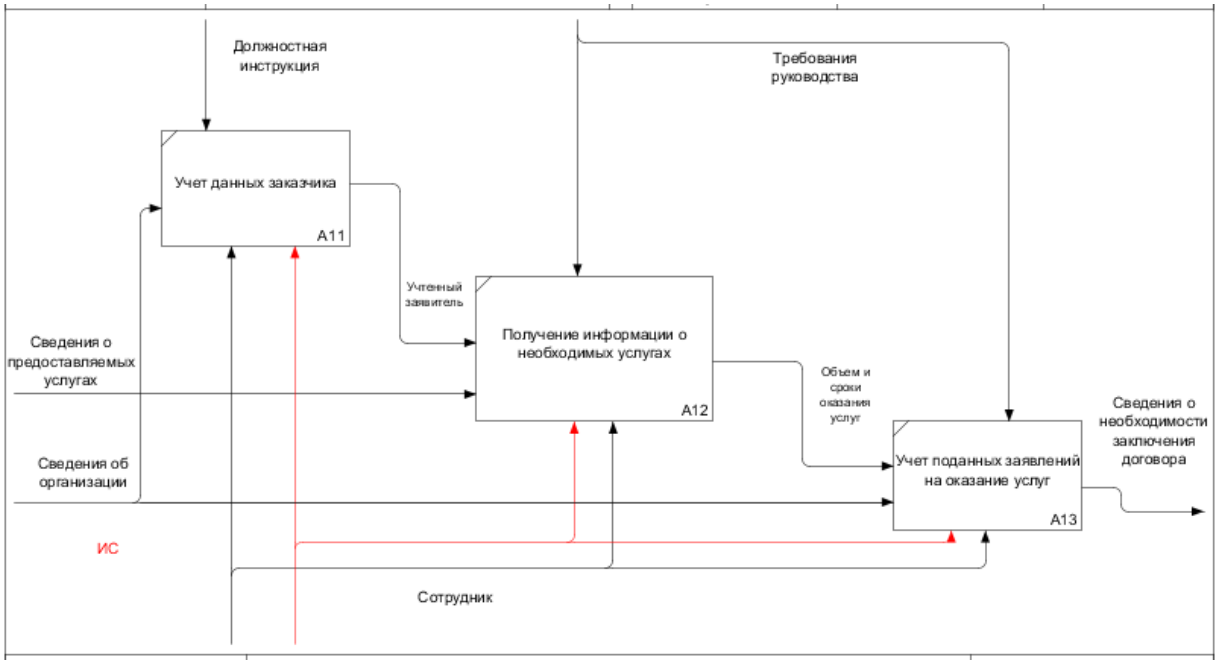

Рисунок 2.3 - Декомпозиции процесса ввода документов

После разработки ИС работы с заявителями будет вестись именно в этой системе. Формирование списка тендеров (заказов) и заявок будет осуществляться после управляющего воздействия оператора системы, то есть сотрудника. Учет заявок также будет осуществляться автоматически. Кроме того, в бедующем появится возможность получение отчетов при переходе на экранную форму с соответствующей формой сортировки, инициализации подготовки отчета и получении его в табличном и графическом виде. Также, появится возможность формирования новых отчетов на основании введенной в систему информации.

2.2 Выбор архитектуры ИС

По способу организации корпоративные информационные системы подразделяются на системы, построенные на основе:

- архитектуры «файл-сервер»;
- архитектуры «клиент-сервер»;
- многоуровневой архитектуры;
- интернет - технологий.

Технология «файл-сервер» – это технология обработки данных, при которой БД размещена на винчестерском диске одного из компьютеров сети в виде файлов. Файл-серверами называют серверы, у которых разделяемым ресурсом является дисковая память (или, говоря иначе файлы, хранящиеся на винчестерском диске) [5].

Архитектура «файл-сервер» предусматривает концентрацию обработки на рабочих станциях (рисунок 2.4).

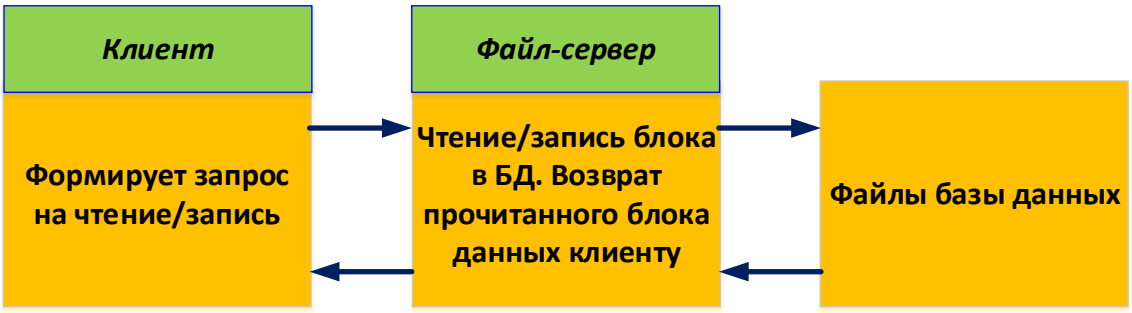

Рисунок 2.4 - Архитектура «файл-сервер»

Достоинства технологии «файл-сервер»:

- простота реализации и дешевизна;
- независимость компьютера от сети;
- высокая защита от несанкционированного доступа.

Недостатки технологии «файл-сервер»:

- не своевременное обновление информации на нескольких ПК;
- высокая стоимость ПК для работы в этой системе;
- сложность изменения структуры данных.

Технология «клиент-сервер» – вычислительная или сетевая архитектура, в которой сетевая нагрузка распределены между поставщиками услуг [6], называемыми серверами, и заказчиками услуг, называемыми клиентами (рисунок 2.5).

Рисунок 2.5 - Технология «клиент-сервер»

Базовый принцип клиент-серверной технологии состоит в делении операций обработки информации на 3 группы, а именно:

- указание и визуализация данных;

- операции обработки информации, присущие для решения задачи из предметной области;

- операции сохранения и контроля данными (БД или файловыми системами).

Исходя из данного разделения, любой техпроцесс выделяет 3 типа программ:

- программы представления, которые исполняют операции 1-й группы;

- программы, исполняющие операции 2-й группы;

- программы доступа к данным, исполняющие операции 3-й группы.

Исходя из этого, выделяют 3 модели представления технологии «клиент-сервер» [5]:

- модель доступности к удаленным данным (Remote Data Access — RDA);

- модель сервера БД (Data Base Server — DBS);

- модель сервера приложений (Application Server — AS).

Базовый принцип клиент-серверной технологии состоит в делении операций обработки информации на 3 группы, а именно:

- указание и визуализация данных;

- операции обработки информации, присущие для решения задачи из предметной области;

- операции сохранения и контроля данными (БД или файловыми системами).

Исходя из данного разделения, любой техпроцесс выделяет 3 типа программ:

- программы представления, которые исполняют операции 1-й группы;

- программы, исполняющие операции 2-й группы;

- программы доступа к данным, исполняющие операции 3-й группы.

Исходя из этого, выделяют 3 модели представления технологии «клиент-сервер» [5]:

- модель доступности к удаленным данным (Remote Data Access — RDA);

- модель сервера БД (Data Base Server — DBS);

- модель сервера приложений (Application Server — AS).

Основные плюсы данной технологии:

- понятная синхронизация данных;

- минимальная цена аппаратуры (производительный должен быть сам сервер);

- быстрое изменение структуры данных.

Минусы данной технологии:

- малая защита от НСД;

- зависимость от ЛВС.

Сетевое многопользовательское ПО разрабатывается по принципу архитектуры «файл-сервер». Данные в виде фала или группы файлов помещены на файловом сервере [12]. Он принимает запросы, которые приходят от клиентов-ПК, и отправляют им требуемую информацию. Но обработка этой информации реализована на компьютерах-клиентах. Любой из ПК запускает полную копию процесса обработки данных. Эта копия независимо управляет самой БД, содержащей данные. Есть лишь одна связь между этими независимыми действиями – файл блокировки, создаваемый для каждого файла БД [12]. И каждая копия реализует изменения индексов, работу с системными таблицами и остальные функции, имеющиеся в компетенции СУБД.

В клиент-серверной архитектуре сервер БД реализует доступ к общим данным и сам выполняет обработку этой информации. Клиент отправляет запросы на чтение или изменение данных на сервер, сформированные на языке SQL. Сервер сам делает необходимые выборки или изменения, контролируя в каждом процессе согласованность и целостность данных и итоговое значение в кода возврата или набора записей отправляет на компьютер клиента.

Минусы файлового сервера заключаются в том, что данные существуют в одном месте, а обрабатываются в другом. При этом их нужно передавать по сети. Это приводит к повышению нагрузки на сеть и к понижению производительности ПО при увеличении числа работающих одновременно пользователей. Другим важным минусом подобной архитектуры является

децентрализованное решение возникающих проблем согласованности и целостности данных и доступа к данным. Данное решение уменьшает надежность приложения.

Архитектура "клиент-сервер" устраняет вышесказанные недостатки и дает возможность распределить всю нагрузку между клиентом и сервером. Это влияет на многие параметры системы: поддержку, производительность и стоимость.

Архитектура «клиент-сервер» бывает и 2-х уровневой, так и 3-х уровневой [15].

В двухуровневой архитектуре присутствуют два элемента: это клиентская ЭВМ и сервер СУБД (рисунок 2.6).

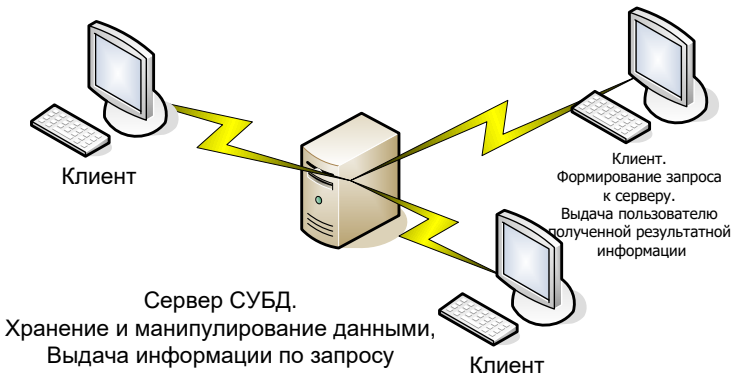

Рисунок 2.6 - Двухуровневая архитектура «клиент-сервер»

Двухуровневая архитектура может быть устроена по-разному [7]:

- «тонкий сервер» - «толстый клиент» (рисунок 2.7);
- «толстый сервер» - «тонкий клиент» (рисунок 2.8).

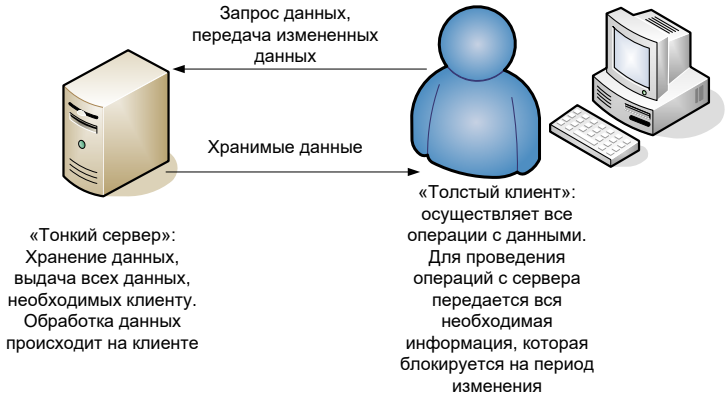

Рисунок 2.7 - Двухуровневая архитектура «тонкий сервер» - «толстый клиент»

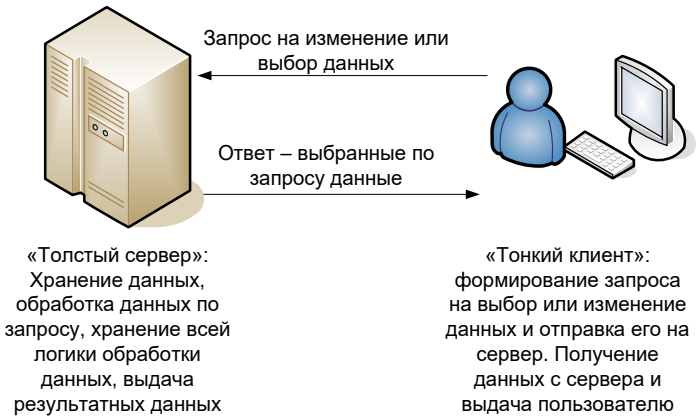

Рисунок 2.8 - Двухуровневая архитектура «толстый сервер» - «тонкий клиент»

Доступ к данным происходит через промежуточный элемент - сервер приложений. Это дает большие преимущества в сравнении с двухуровневой архитектурой при построении крупномасштабных ИС на различных аппаратных платформах при большом количестве клиентов и их территориальной распределенности. Такая архитектура обеспечивает дополнительную надежность и отказоустойчивость системы, неограниченно увеличивать мощность системы и количество клиентов благодаря использованию нескольких серверов приложений. При этом становится возможным перенос БД с одной СУБД на другую и изменения логики изменения данных без необходимости обновления клиентских программ. Сервер приложений не только выполняет предметные задачи, но и уменьшает нагрузку на сервер БД.

Архитектура системы автоматизации по трехуровневой схеме «клиент-сервер» представлено на рисунке 2.9.

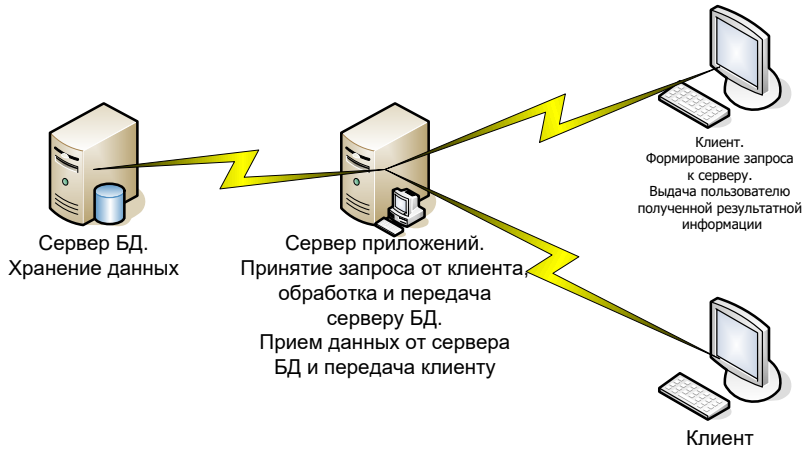

Рисунок 2.9 - Трехуровневая архитектура «клиент-сервер».

Наиболее приемлемой для решения текущей задачи является выбор двухуровневой архитектуры «тонкий клиент» - «толстый сервер», потому, что преимущества трехзвенной архитектуры не являются необходимыми для выбранной задачи. Стоимость и сложность организации трехзвенной архитектуры выше, нежели у двухуровневой. Существующая информационная инфраструктура подходит именно для двухуровневой архитектуры [9].

Перед проектированием АИС важно подробно обосновать актуальность ее создания, четко описать цели и задачи проекта, возможную прибыль, затраты времени, ресурсы и ограничения. Такая работа часто зовется стратегическим планированием ИС, и для ее реализации назначается менеджер проекта [10].

2.3 Выбор стратегии автоматизации рынка инновации

Одним из важнейших этапов всей предпроектной подготовки является выбор стратегии автоматизации. Из-за неправильного выбора система может как замедлить производство, так и не принести ожидаемой прибыли.

Базовые принципы стратегии автоматизации:

- цель – последовательность и сферы деятельности организации, в которой они будут автоматизироваться;

- способы реализации – применение комплексной автоматизации, а также по участкам и направлениям;

- техническая политика – внутренние стандарты, поддерживаемые организацией;

- возможные ограничения – временные, финансовые и т.п.;

- реализация процедуры управления изменениями в планах.

План автоматизации должен соответствовать основным задачам деятельности организации. При составлении плана должны учитываться следующие факторы:

- средний период изменения технологий производства;

- среднее время жизненного цикла продукции организации;

- план развития самой организации;

- изменения в функциях персонала.

Необходимо понимать, что автоматизация – это один из способов достижения бизнес-целей. В любом плане автоматизации всегда есть главная цель – миссия, направление деятельности и модель бизнеса. Поэтому план автоматизации – это всего лишь инструкции со своими сроками и видами деятельности.

Необходимо учитывать и основные ограничения при выборе плана автоматизации, а именно:

- временные ограничения;

- финансовые ограничения;

- технические ограничения;

- человеческий фактор.

Финансовые ограничения определяются суммой вложений, которые организация может внести в развитие автоматизации.

Временные ограничения связаны со следующими причинами:

- обучением сотрудников;

- временными затратами на внедрение системы.

Ограничениями, связанными с человеческим фактором, являются:

- особенности трудового законодательства и рынка труда;

- отношение самого персонала к процессам автоматизации.

В настоящее время выделяют две главные стратегии оптимизации: подгонка уже действующего программного продукта под определенный бизнес-процесс или реорганизация всего бизнес-процесса и автоматизация уже вновь созданной структуры. Выбор плана оптимизации в большинстве случаев зависит от перспектив развития организации-заказчика и ее экономических возможностей.

Существует несколько направлений оптимизации.

Кусочная (хаотичная) автоматизация. Она представляет собой один из неэффективных методов вложения средств для подъема организации. Такой подход характеризуют необходимые на данный момент внедрения информационных технологий, что не соответствует реальным потребностям организации. В этом случае само предприятие получит прикладную систему, но ее стоимость будет сопоставима с качественным комплексным решением. Эту стратегию выбирать можно в том случае, если нужно сэкономить средства и время на внедрение системы. Но вся дальнейшая модернизация будет стоить намного дороже и станет нерациональна и экономически невыгодна.

Следующий вид автоматизации – по участкам. Она включает в себя обновление отдельных производственных или управленческих линий, которые сопоставимы по функциональным особенностям. Этот вид автоматизации можно применим в том случае, если у организации нет дополнительных средств на автоматизацию в полном объеме, а автоматизированные участки дадут хороший экономический рост, поэтому применяется она часто на производственных участках.

Следующий вид автоматизации – по направлениям. Она предполагает полную автоматизацию отдельных видов деятельности организации. Главной отличительной особенностью от автоматизации по участкам является то, что

вся деятельность затрагивает все подразделения и службы, задействованные в автоматизируемом направлении. Заключительным этапом этого метода является полная автоматизация компании.

Последний вариант направления – полная автоматизация. Она предполагает систему, которая состоит из множества элементов разного уровня и значения. В нее как правило обычно входят другие подсистемы, модули, функции, блоки управления, текущие задачи и т.д. Это построение позволяет использовать один и тот же алгоритм для просчета разных задач, исключает дублирование данных из разных источников. Минусом полной автоматизации являются большие затраты на реализацию такой схемы, и высокий уровень первоначального планирования и расчетов.

Проанализировав данные виды стратегий, лучшей была признана стратегия по направлениям, так как это наиболее подходящий вариант для рассматриваемой организации.

2.4 Обоснование проектных решений по программному обеспечению

2.4.1 Выбор языка программирования для разработки информационной системы

Необходимо определить язык программирования, который поможет создать приложение для взаимодействия с БД.

При формировании программного комплекса следует рассматривать его как отложную систему, разные части которой взаимодействуют друг с другом для решения общей задачи в то же время, различные его программные модули могут решать независимые задачи. При проектировании должны программные систем необходимо поддерживать такого рода модульность, как для разделения труда программистов, так и доя упрощения поддержки конечного продукта и расширения его функционала.

Разделяя систему на модули с определенным функционалом, скрытым за некоторым интерфейсом, появляется возможность разрабатывать различные части программной системы1, используя независимым стек технологий. Поэтому нет необходимости писать всю систему целиком, используя один язык программирования. Необходимо рассматривать язык программирования как инструмент и, соответственно использовать тот инструмент, которым лучше подходит к решаемой задаче ключевым параметрами при выборе языка программирования являются:

- кроссплатформенность,
- торговое мультипарадигменность,
- эффективность,
- безопасность,
- выразительность,
- поддержка,
- расширяемость,
- поддержка методов параллельного программирования.

Кроссплатформенность - обеспечивает возможность сборки исходного кода без его изменения доя различные аппаратные и программные систем. Под аппаратными системами в первую очередь понимаются системы1 с различными архитектурами процессора и объемом оперативной/постоянной памяти. Однако наличие одинаковой аппаратной платформы1 еще не гарантирует совместимость программного обеспечения. Как правило, современное программное обеспечение разрабатывается с учетом работы на этой аппаратной платформе некоторой операционной системы. Для языка программирования кроссплатформенность означает наличие компилятора дои среды выполнения под различные аппаратные и программные платформы^

Торговое мультипарадигменность - поддержка языком различные парадигм программирования, что позволяет писать код продукта в том ключе, который более удобен доя конкретно решаемой задачи

Так, например, для программирования общей архитектуры системы хорошо подходит ООП парадигма, а код алгоритмов часто просто приходиться писать в императивном стиле на языке низкого уровня, пожертвовав удобством ради скорости выполнения.

Эффективность

Эффективность выполнения одной и тот же последовательности операций написанного на различных языках крайне трудно оценить, как минимум из-за различия в семантике языка, однако можно с уверенностью разделить языки по скорости выполнения генерируемого на их основе машинного байт кода на три класса.

Компилируемые языки

Это языки компиляция которых порождает объектный код целевой платформы. Такой объектный код являет непереносимым, в том плане, что он предназначен для исполнения на процессоре той архитектуры, для которых компилировала. Таким образом, для каждой аппаратной платформы может быть собран объектный код на основе программного кода продукта компилятор данного языка в объектный код этот аппаратной платформы. Это позволяет.

Компилируемые языки работают быстрее интерпретируемых и комбинирующихся в байт-код.

Языки с компиляцией в байт-код

Такие языки транслируются в промежуточное представление - байт код виртуальной машины, который можно рассматривать как аналог ассемблера для целевой виртуальной машины.

Интерпретируемые языки

Программный код написанной на этот категории языков работает медленнее всего. Причина в том, что в случае компиляции в машинный байт код компилятор строит абстрактное синтаксическое дерево на основе всей информации о программе, которое анализируется и затем оптимизируется. Интерпретатор лишен возможности проанализировать весь программный код

целиком, он вынужден анализировать команды построчно и интерпретировать их, каждую в отдельности.

Безопасность

Многие конструкции языка могут содержать потенциальные опасности их использования. Возможности низкоуровневого доступа к памяти, преобразование типов операции с указателями, отсутствие обработки исключений, и встроенной проверки критических ситуаций, такта как выход за границы массива обращение к неинициализированным данным приводят к тому, что код становится небезопасным. Во время выполнения такой код может выполнить недопустимую операцию дои привести к краху ОС Для параллельных программных систем факторами, уменьшающими безопасность кода, является отсутствие механизмов синхронизации потоков разделяемых переменных, атомарных операций и других языковых средств

Выразительность языка

Язык накладывает множество ограничений при проектировании архитектуры ПО. Языковые конструкции одновременно ограничивают программиста и в то же время предоставляют ему некоторый уровень абстракции. Так, например, программы, написанные в оперативном стиле на ассемблере, предоставляют неограниченный контроль над машиной, но в то же время, код на языке ассемблера труден доя восприятия человеком, его сложно модифицировать и поддерживать. Это происходит по причине того, что ассемблерные инструкции описывают элементарные операции над данными, в то время как программист представляет логику работы более абстрактно.

За структурным программированием появились объектно-ориентированное и др. При всем этом разнообразии рассматриваемые далее языки за исключением С, относятся к объектно-ориентированному типу, но при этом поддерживают и оперативную и функциональную парадигмы, однако большинство кода на таких языках пишется в объектно-ориентированном стиле. ООП парадигма прочно задала свое место в разработке программных систем благодаря основным принципам: наследованию, инкапсуляции, полиморфизму

и абстрагированию. Так же как все структурные языки имеют реализации цикла, ветвления, так и все ООП языки так или иначе реализует языковые средства доя организации сокрытия функционала за некоторым интерфейсом, полиморфного поведения объектов языке и обеспечения низкой связанности компонентов. Однако все языки делают это по- своему.

В современном мире у программистов широкий выбор средств программирования: множество различных языков, как универсальных, которые используются в разных сферах программирования и способны работать на нескольких аппаратных и программных платформах, так и специализированные, направленные на определенную область разработки, такие как, PROLOG - язык программирования высокого уровня. Он используется для реализации систем и программ искусственного интеллекта. Существует много разных классификаций языков программирования: по уровню семантики (языки высокого и низкого уровня), по используемым парадигмам программирования, по безопасности и т.д. Предъявляемые требования к разрабатываемой системе определяют выбор средств ее реализации на этапе проектирования.

Сравнение свойстваязыков Python и PHP

Одним из распространенных средств разработки является высокоуровневый язык программирования Python. Python имеет поддержку различных парадигм программирования (структурного, объектно-ориентированного функциональное и других); в нем используется динамическая типизация переменных, обеспечивается периодическое освобождение памяти от неиспользуемых объектов.

Язык Python обладает развитой системой модулей, как стандартных, также написанных на Python, а, значит, обладает такими же преимуществами, как кроссплатформенность, и позволяет реализовывать наиболее общие задачи, так и специфические.

Из-за функциональных возможностей Python может использоваться в различных сферах: разработка web-приложений, автоматизированных ИС, научных вычислительных комплексов, графических пакетов.

PHP – это язык программирования общего назначения, используется в основном при работе над web-приложениями, для генерации HTML-страниц и работы с базами данных. PHP имеет Си-подобный синтаксис и является парадигмальным и кроссплатформенным, ядро PHP реализует средства для автоматического управления памятью; после выполнения работы скрипта вся выделенная память возвращается системе [2].

Динамические библиотеки PHP предоставляют широкие возможности для работы с базами данных, поддерживается DBX для работы на абстрактном уровне, стандарт ODBC; осуществляется коммуникация с использованием различных протоколов (IMAP, SNMP, POP3, HTTP и другие); также PHP обеспечивает работу с сокетами, динамической графикой, криптографическими библиотеками. [3]

Основное применение PHP – разработка web-приложений. Много web - фреймворков и CMS-систем созданы при помощи PHP.

Языки Python и PHP имеют схожие исходные характеристики. Необходимо рассмотреть их возможности при реализации конкретной задачи.

По быстродействию программа, написанная на Python, уступает программе, написанной на PHP в 3,5 раза, но загружает память в 1,5 раза меньше. Приведенные данные носят приблизительный характер из-за возможных погрешностей измерения характеристик при использовании методов с ограниченной точностью. Но они позволяют дать общее представление о производительности кода и в соответствии с поставленными задачами и требованиями, предъявляемыми к ним и сделать выбор в сторону того или иного языка. Можно выделить, что PHP широко используется в web-программировании, из-за скорости выполнения его скриптов. В целом полученные результаты соответствуют известным раннее выводам о характеристиках данных языков.

Следовательно, для реализации, рассмотренной ИС выбираем язык программирования PHP.

Одним из основных языков web - программирования является PHP. Его главные преимущества: понятный синтаксис, быстродействие, работа с множеством хостингов, на PHP созданы многие используемые движки.

Другой распространенный язык веб программирования на платформе Unix - язык Perl. Он сложный, его синтаксис запутанный, и изначально не был создан для web - программирования.

Поэтому в рамках языка разработки соединения с БД выбираем PHP.

PHP был выбран, в связи с тем, что:

- модули PHP запускаются из области памяти, выделенной программе самой ОС. ASP подгружает для действия свои модули COM, чем сильно забивает ОЗУ и CPU;

- объединение с PHP с этой СУБД MySQL более адекватна, чем у ASP. Есть много утилит на PHP для работы с БД MySQL, где представлен весь набор свойств в сравнение с другими БД. Тут имеются очень полезные встроенные функции, которые недоступны для остальных БД. Основным достоинством PHP считается поддержка широкого круга БД: Oracle, Microsoft SQL server, MySQL и т.д.;

- плюс PHP - это неимение временных проблем с исправлением отдельных ошибок, что позволяет быстро реагировать и корректировать доработки.

2.4.2 Выбор системы управления базами данных

Сейчас реляционная модель данных становится базовой при построении БД потому, что организация хранения данных в виде таблиц логична для пользователя. Эта особенность приводит к множеству реляционных СУБД. Выбор самой удобной СУБД – сложная задача.

Опишем самые распространенные СУБД.

Oracle Database

Объектно-реляционная СУБД. Самая первая версия стала коммерческой СУБД, поддерживающей SQL-язык. В настоящее доступно 6 версий СУБД:

- Enterprise Edition: Полноценная версия без ограничений. Этот продукт необходим для крупных предприятий. Пользователям предоставляются опции, помогающие архитектурно и функционально оптимизировать сервер;

- Standard Edition: Есть ограничение по числу процессорных разъемов (не более 4-х). Эта редакция необходима для организаций среднего размера или в отделе крупной организаций;

- Standard Edition One: Есть ограничение по числу процессорных разъемов (не более 2-х) и нет поддержки кластеризации;

- Personal Edition: Она рассчитана на однопользовательский режим;

- Lite: Версия используется в мобильных и встраиваемых устройствах и применяемся нп малых предприятиях;

- Express Edition (XE): Доступная СУБД. Используется 1 процессор, есть ограничения по объему ОЗУ (1 Гб) и объему БД (11 Гб) [13].

Вышеперечисленные версии СУБД имеют похожий исходный код и аналогичный функционал, кроме отдельных узконаправленных методик конкретных версий. Основная задача стандартной, персональной и мобильной версий - понижение стоимости владения, легкость использования ПО.

Oracle Database считается кроссплатформенным ПО, так как около 80 % программного кода написано на языке Си. Ядро сервера, с остальными 20 % кода, переделывается под необходимую платформу, поскольку создано на машинно-зависимых языках [14].

Microsoft Access

Реляционная СУБД, созданная Microsoft, имеющая встроенный Visual Basic for Applications (VBA), позволяющий создавать приложения в Access для работы с БД. Помимо VBA в приложении применяется язык структурированных запросов SQL и макрокоманды [15].

MS Access считается файл-серверным СУБД - это уменьшает круг её использования. В роли движка БД стоит Access Database Engine или Microsoft Jet 4.0 в зависимости от ревизии СУБД [1]. MS Access может совмещаться с внешними СУБД клиент-серверной архитектуры, например, с MySQL, Firebird, Oracle и др. Благодаря автоматическому резервированию после перехода к следующей записи СУБД устойчив к сбоям в электропитании. Несмотря на наличие открытых версий, программный комплекс лучше всего применять после приобретения лицензии.

Проект MS Access сохранен в файле формата accdb, что упрощет его передачу и работу с программой. Персоналу, не имеющему достаточный уровень знаний взаимодействия с данной СУБД, помогают различные конструкторы. Плюсом такой настольной СУБД становятся русифицированный интерфейс и хорошая СЗИ.

MS SQL Server

Первая версия СУБД стала совместной работой фирм Sybase, Ashton-Tate и Microsoft [16]. MS SQL Server принадлежит к СУБД клиент-серверной архитектуры. Есть большое количество версий и обновлений. Последние версии ПО уже включают компонент ядра СУБД (Database Engine); набор технологий для репликации с БД, службы для работы с данными - предоставление отчетов, анализ данных и т. п. [17].

Microsoft SQL Server Express Edition — это версия продукта, уменьшенного по функционалу, но находящаяся в открытом доступе [18]. Данная версия применяется в рамках малой компании из-за уменьшенных возможностей в сравнении с MS SQL Server. Языком просмотра выступает процедурное расширение языка SQL — Transact-SQL.

Microsoft SQL Server совместим с БД других форматов, например: Oracle, DB2, Sybase и Microsoft Access [19]. Эта СУБД имеет простой доступ пользователей к изучаемой информации, что реализовано методом объединения с пакетом программ Microsoft Office. Дополнительная защита данных реализуется возможностью шифровать БД, файлы журналов или файлы данных.

Sybase A daptive Server Enterprise

Реляционная СУБД, создана фирмой SAP [20]. Первоначально была разработана вместе с Microsoft SQL Server, и поэтому применяет Transact - SQL как базовый язык запросов. Является СУБД клиент-серверной архитектуры. Применяется в системах средних или больших организаций. Определяется удобством использования и небольшой ценой для части рынка СУБД, поддерживающих большие БД. Используется для взаимодействия с большим числом пользователей, объемами данных для важных объектов. Данная СУБД не зависит от ПО других разработчиков.

SAP Sybase Adaptive Server Enterprise Cluster Edition — версия СУБД имеет усиленную отказоустойчивость, которая обеспечивается разбиением на кластеры и переводом пользователей с отказавшего узла кластера на рабочий [21]. Операция выполнится повторно при сбое в процессе реализации транзакции по факту перехода на рабочий узел.

MySQL

Универсальная СУБД. Находится в открытом доступе, но имеет коммерческую лицензированную версию от MySQL AB. с открытым кодом, первоначально созданной шведской компанией MySQL AB, а с 2008 - 2010 года разрабатывалась фирмой Sun Microsystems [12]. В настоящее время разработка и поддержка ПО лежит на Oracle. Начальная версия СУБД вышла в 1995 году, а представили ее в январе 2001 года. Эта СУБД используется для управления малых и средних систем. СУБД MySQL применяется там, где есть клиент-серверная и встроенная архитектура. В версиях выше MySQL 3.22 все ограничения с таблиц сняты [23].

Для упрощения работы с БД MySQL существует визуальный интерфейс — PhpMyAdmin [11]. СУБД MySQL версий 5.0 и выше отвечает стандарту структурированного языка запросов SQL, поэтому она совместима с многими версиями БД. Главный язык разработки - C/C++.

На основании проведенного анализа выбираем язык программирования php и СУБД Mysql.

Проведенный во второй главе выпускной квалификационной работы анализ деятельности рассматриваемой организации позволяет сделать вывод о том, что описанные бизнес-процессы менеджеров по работе с клиентами в настоящее время используют неэффективно рабочее время сотрудников компании, и что внедрение информационной системы позволит данный недостаток устранить. На основании исследований порядка работы с клиентами, принятого в данной компании, сформулирован ряд требований к разрабатываемому программному обеспечению, в результате определен его вид как веб-приложение, что позволит с минимальными потерями внедрить такое программное обеспечение в бизнес-процессы туристического агентства без нарушения их порядка и обеспечить удобство его использования. Также для снижения трудоёмкости разработки приложения и сроков его разработки выбраны наиболее распространённые язык программирования PHP и система управления базами данных Mysql.

2.5 Обоснование проектных решений по техническому обеспечению.

Техническое обеспечение (ТО) состоит из персонального компьютера, оргтехники, средств связи, сетевого оборудования. Влияние на подготовку, обработку и передачу данных влияет вид информационной технологии, который основан на технической оснащенности (ручной, автоматизированный, удаленный).

Система технических средств включает в себя:

- ПК;

- устройства хранения, механизмы сбора, обработки, передачи и вывода данных – сканеры, жесткие диски, принтеры, съемные носители, факсы;

- устройства для передачи данных и линии связи – коммутаторы, маршрутизаторы, модемы;

- расходные материалы – CD (DVD)- диски, бумага, и т. д.

Для правильного выбора ПК, необходимо руководствоваться рядом характеристик, к которым можно отнести цену, надежность, удобство использования, производительность и др. Успех разработки системы и возможность работы с необходимым ПО зависит от величины каждого параметра.

Для любого элемента данной схемы имеется перечень критериев, важных при реализации выбора ТО. Критерии можно представить так:

- частота работы ЦПУ;
- максимальное разрешение экрана монитора;
- установленный объем ОЗУ.

Для выполнения поставленной задачи придется использовать ПК уровня вычислительной мощности не менее AMD 2000 Мгц, либо Intel 2000 Мгц, с установленной ОЗУ свыше 2 Гб. С системой проще всего работает в экранном разрешении 1024x768 на экране диагональю не ниже 17 дюймов. Данные технические средства имеют оптимальную конфигурацию для выполнения всех требуемых задач.

Технические характеристики подходящих рабочих станций приведены в таблице 2.1:

Таблица 2.1 - Характеристики рабочих станций

Модель процессора	Intel Core i3-370M 2.4GHz	Intel Atom Single Core N270 - 1.66GHz
Оперативная память	3072Mb	2048Mb
Жесткий диск	320Gb	160 Gb
Оптический привод	Нет	DVD-RW
Видеоадаптер	Интегрирован в чипсет	Интегрирован в чипсет

В таблице представлены характеристики рабочих станций, но наиболее подходящая будет вторая станция, т.к. в ней есть оптический привод.

Для сканирования, печати и копирования документов необходимо наличие соответствующего оборудования. При аварийном отключении

электропитания в целях сохранения данных персональный компьютер должен быть оборудован блоком бесперебойного питания(ББП).

ББП выбирается исходя из мощности, потребляемой компьютером, и из требуемого времени работы компьютера. Наиболее характерными значениями являются 600-800 Вт и 5-10 минут. Под эти параметры подходит ИБП APC Smart-UPS 750VA, характеристики приведены в таблице 2.2.

Таблица 2.2 -Технические характеристики ИБП APC Smart-UPS 750VA

Характеристика	Значение
Максимальная выходная мощность	500 Ватт / 750 ВА
Максимальное задаваемое значение мощности	500 Ватт / 750 ВА
Номинальное выходное напряжение	230V
Номинальное входное напряжение	230V
Входная частота)	50/60 Hz +/- 3 Hz (auto sensing
Тип входного соединения	IEC-320-C14 inlet
Диапазон входного напряжения при работе от сети	160 - 285В
Диапазон регулировки входного напряжения при работе от сети	151 - 302В
Типовая продолжительность работы в автономном режиме под половинной нагрузкой	16.4 Минуты (250 Ватт)
Типовая продолжительность работы в автономном режиме под полной нагрузкой	4.8 Минуты (500 Ватт)

Выводы по второму разделу

Во втором разделе ВКР разработаны модели бизнес-процессов работы с заказчиками как будет, произведено обоснование проектных решений по техническому обеспечению и программному, выбраны стратегия автоматизации и архитектура системы, что позволяет перейти непосредственно к разработке информационной системы электронного рынка инноваций.

3 Описание реализации информационной системы

3.1 Описание реализации системы

Проект разработан на PHP 5.4.39 с использованием фреймворка «Yii 2.0.15 Basic». Для хранения данных используется база данных MySQL.

Для каждой таблицы используемых базы данных создается модель (Model из Model-View-Controller). Все модели нашего проекта располагаются в директории «basic\models».

Большинство контроллеров нашего проекта реализует CRUD - (англ. create, read, update, delete — «создание, чтение, обновление, удаление») сокращённое именование 4-х базовых функций, используемых при работе с наборами данных.

Консоль управления Yii2 позволяет для конкретной модели выполнить генерацию CRUD. Генератор CRUD создает контроллер, набор действий контроллера («Index», «View», «Create», «Update», «Delete») и ряд видов («View») соответствующих действиям.

Контроллеры нашего проекта находятся в папке «basic\controllers».

Представления, соответствующие действиям контроллеров, располагаются в папке «basic\views».

Таблица 3.1 - Описание базовых действий контроллеров

Название действия	Описание действия
Index	Метод просмотра списка всех записей
View	Метод просмотра записи
Create	Метод добавления записи
Update	Метод редактирования записи
Delete	Метод удаления записи

Работа пользователя с системой осуществляется через обращения к действиям контроллеров.

Пример обращения пользователя к действию «view» (просмотр) контроллера «Tender» (Заказ): http://localhost:7082/basic/web/tender/view/1

При обращении к действиям контроллеров пользователь обращается к действию системы осуществляется через GET запрос протокола HTTP. Ядро Yii2 выполняет интерпретацию данного запроса, передает запрос в соответствующий контроллер, контроллер проверяет права доступа пользователя для обращения к действию, выполняет действие (Index, View, Create и т.д.) и выдает пользователю результат.

Реализованная база данных показана на рисунке 3.1.

Рисунок 3.1 - Схема базы данных

Она была разработанная в программе dbForge Studio for MySQL.

3.2 Описание контрольного примера реализации проекта

Для использования приложения необходимо перейти по ссылке. В качестве главного окна программы показывается список новостей по теме инноваций.

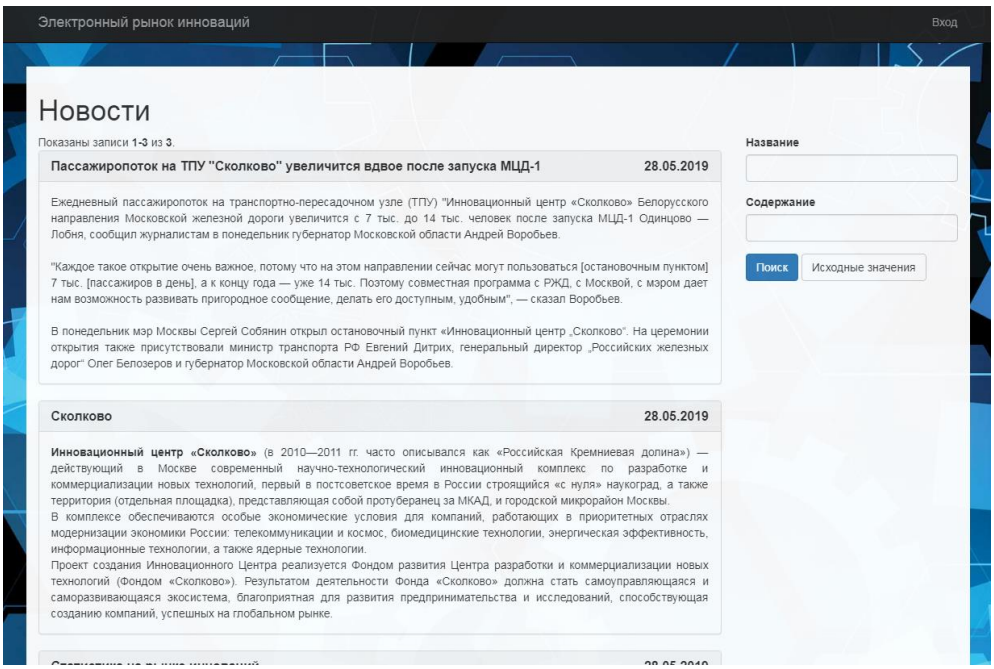

Рисунок 3.2 - Главная форма программы, список новостей

На главной форме поддерживается форма поиска новостей по ключевым словам.

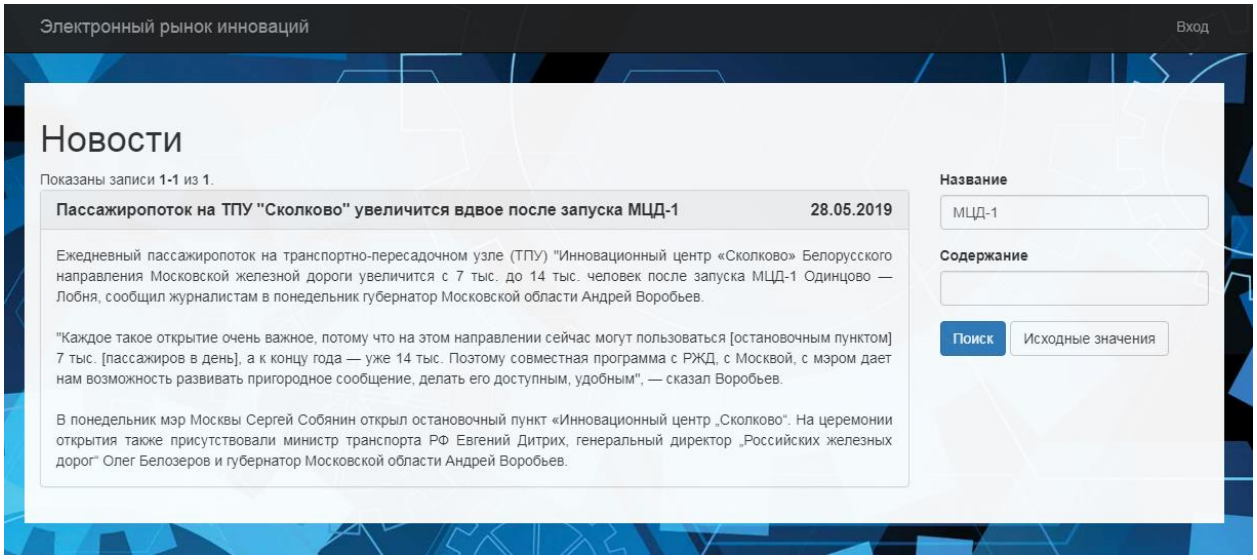

Рисунок 3.3 - Поиск новостей

Рассмотрим функции администратора системы. Для входа в панель управления необходимо авторизоваться.

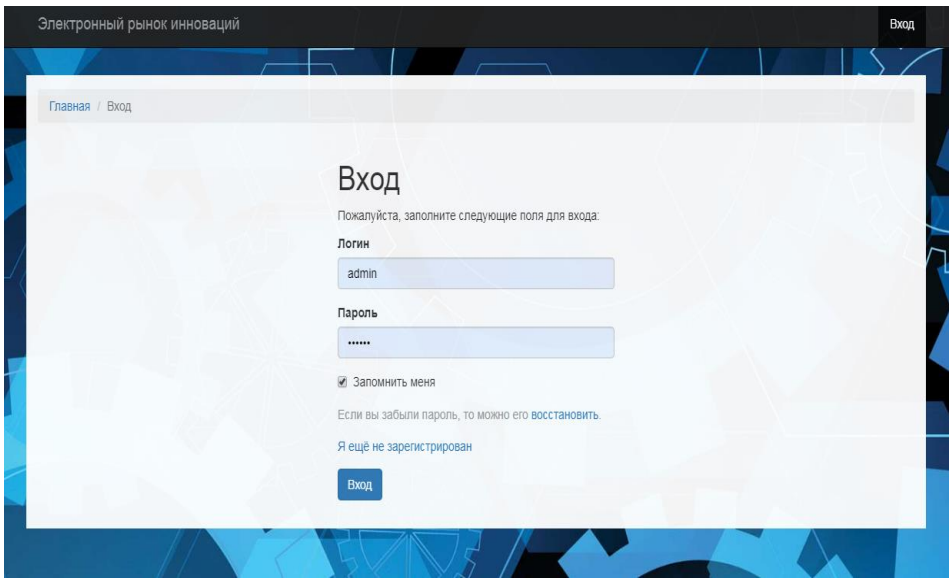

Рисунок 3.4 - Форма авторизации

После входа в панель управления администратор видит список поступивших заказов.

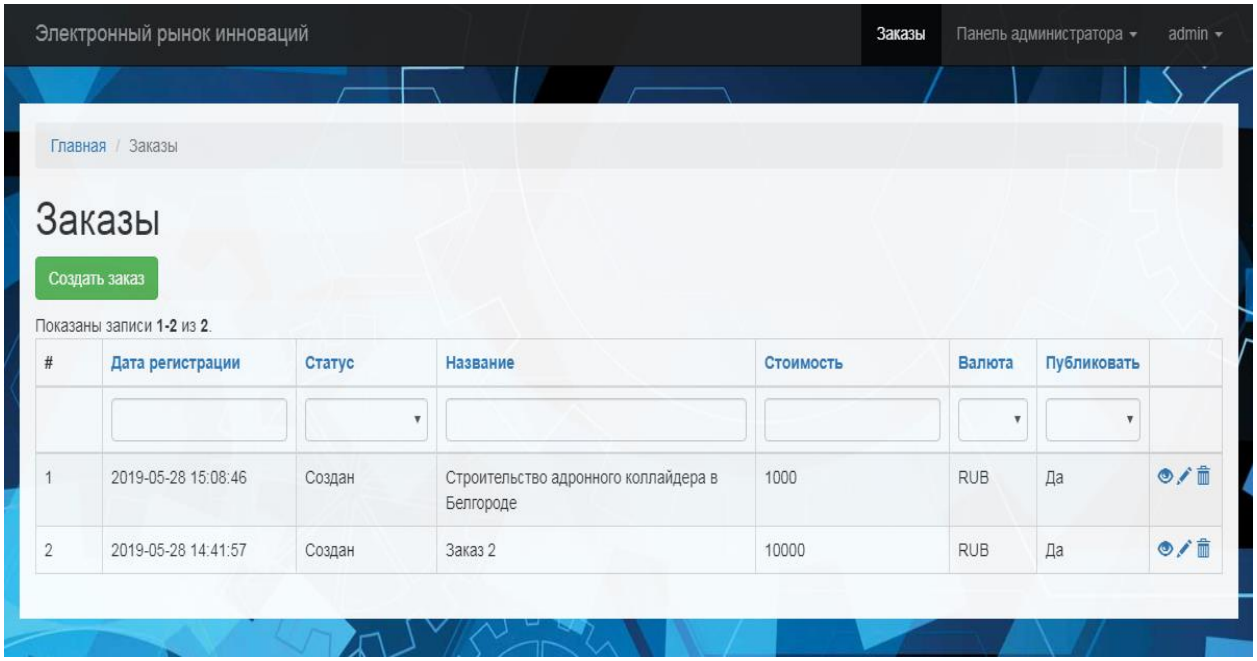

Рисунок 3.5 - Список заказов

В списке приводится дата регистрации заказа, его статус, название, стоимость, валюта и статус публикации – администратор может скрыть заказы от публикации.

Администратор может получить доступ к подробностям заказа, где приведены все его данные.

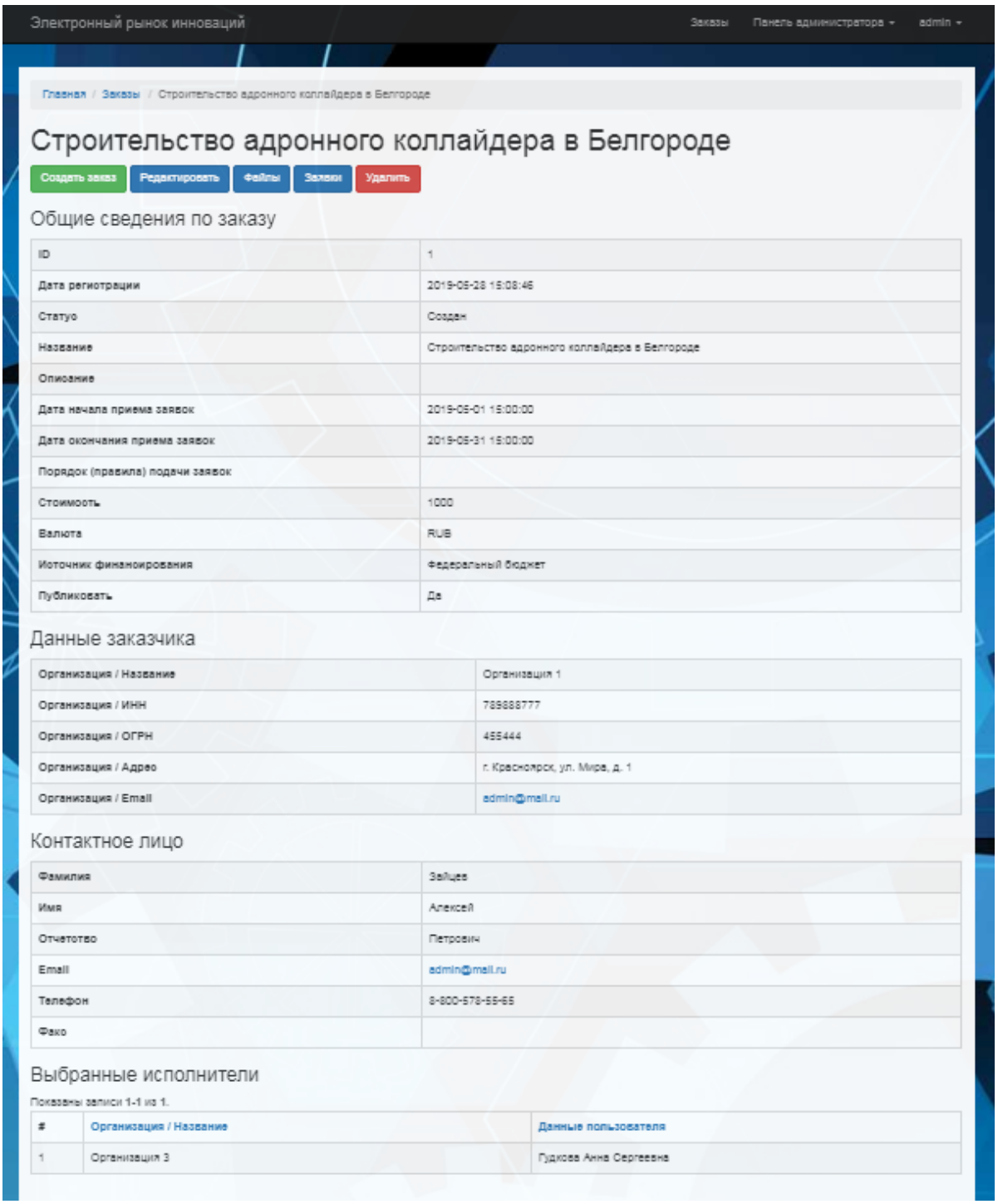

Рисунок 3.6 - Описание заказа

На каждый заказ могут подаваться заявки на его выполнение.

В данной форме указывается период приема заявки, источники финансирования, данные о заказчике и исполнители проекта.

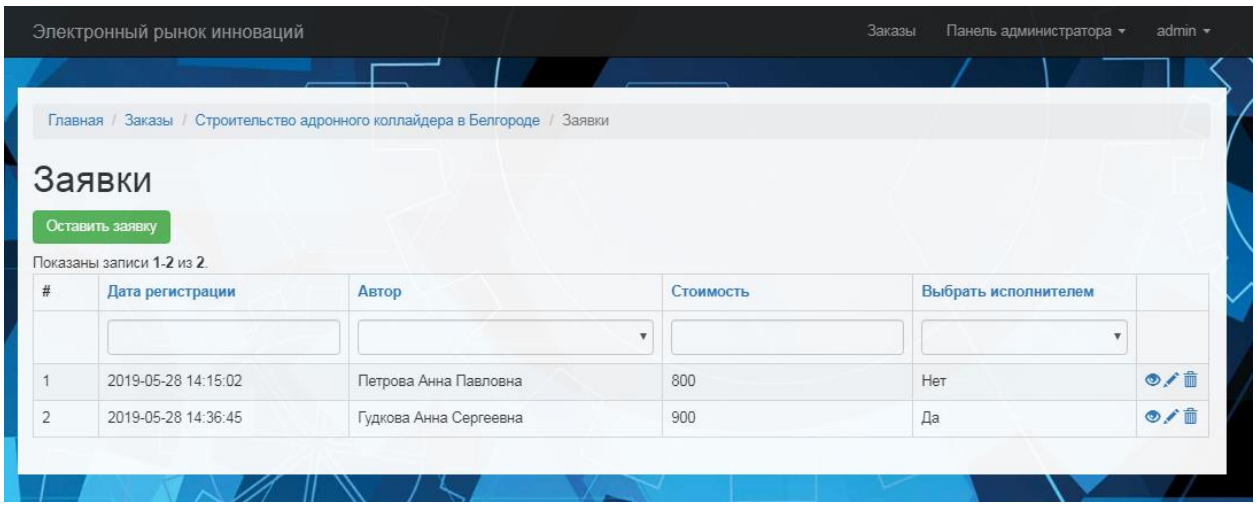

Рисунок 3.7 - Заявки на выполнение заказа

В описании заявки приводится дата регистрации заявки, предложение стоимости, данные заявителя, а также данные контактного лица.

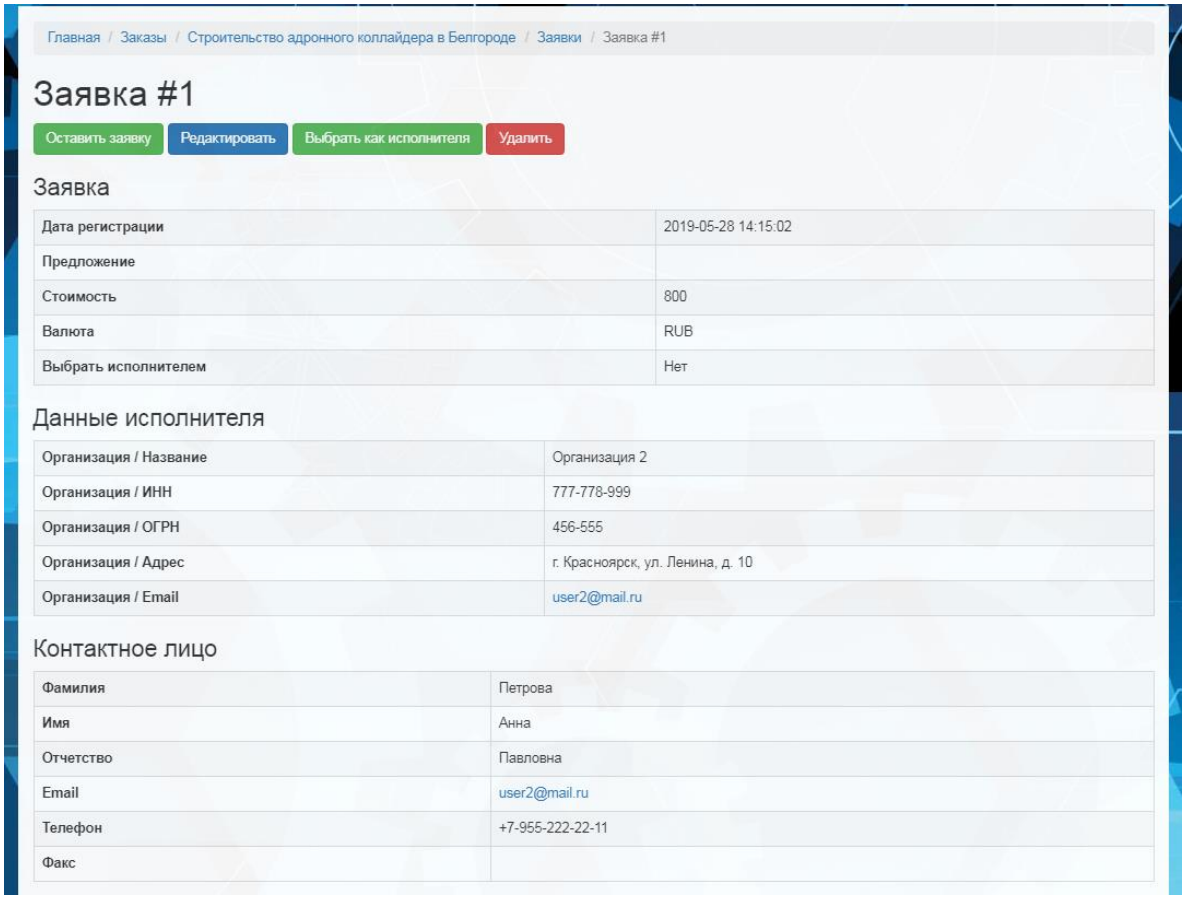

Рисунок 3.8 - Данные заявки

К каждой заявке могут быть приложены файлы

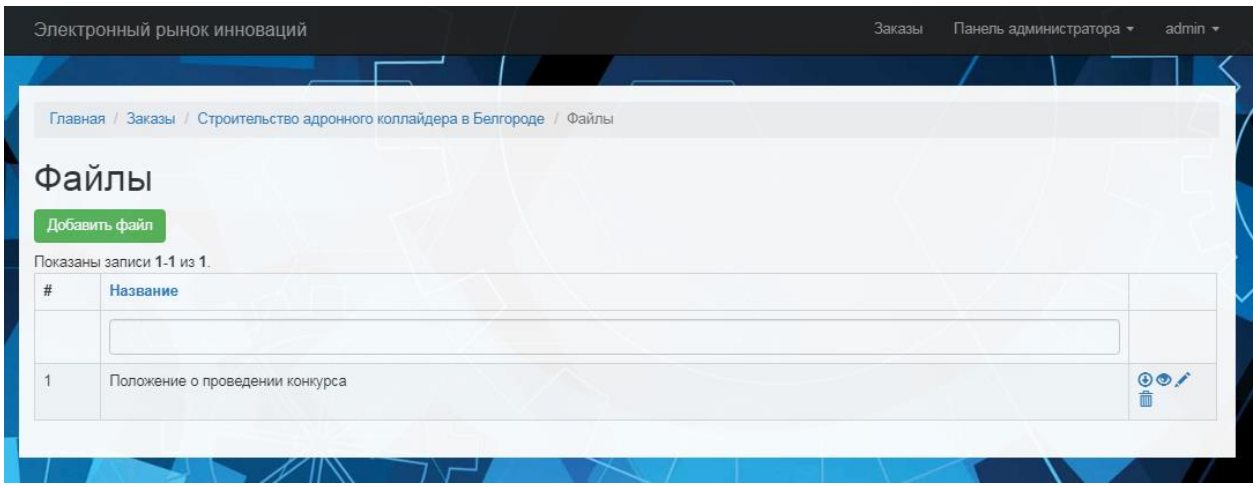

Рисунок 3.9 - Файлы заявки

Администратор может управлять таким файлами, удаляя, редактируя, просматривая их.

Также администратор управляет пользователями.

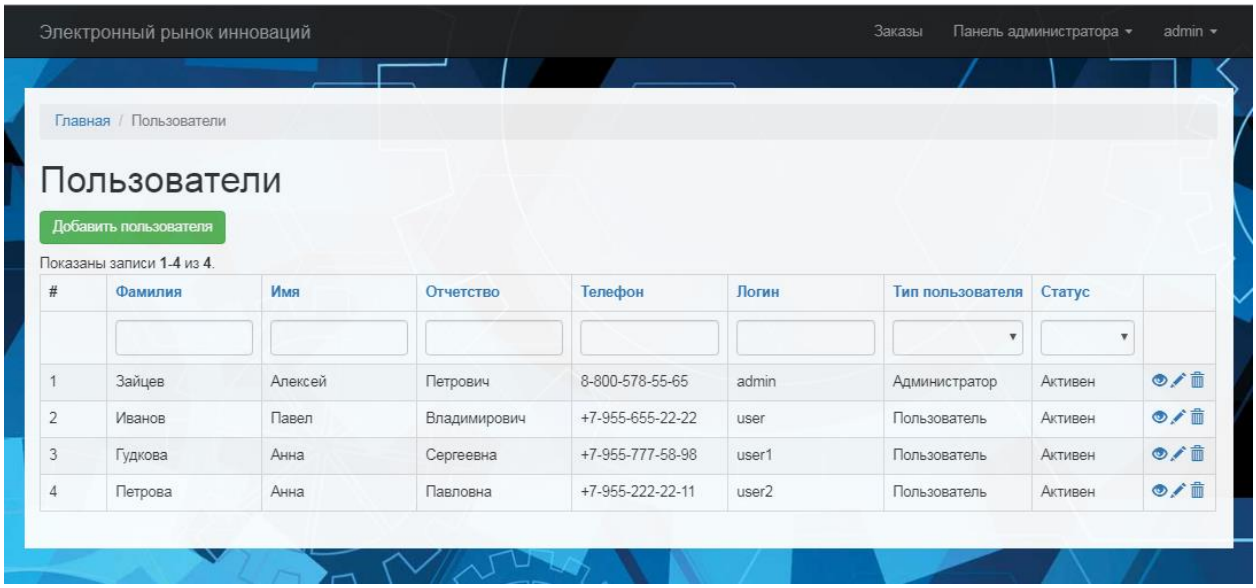

Рисунок 3.10 - Пользователи

Для того, чтобы подать заявку на выполнение заказа, пользователь регистрируется на сайте. После чего, пройдя авторизацию, видит список актуальных заказов.

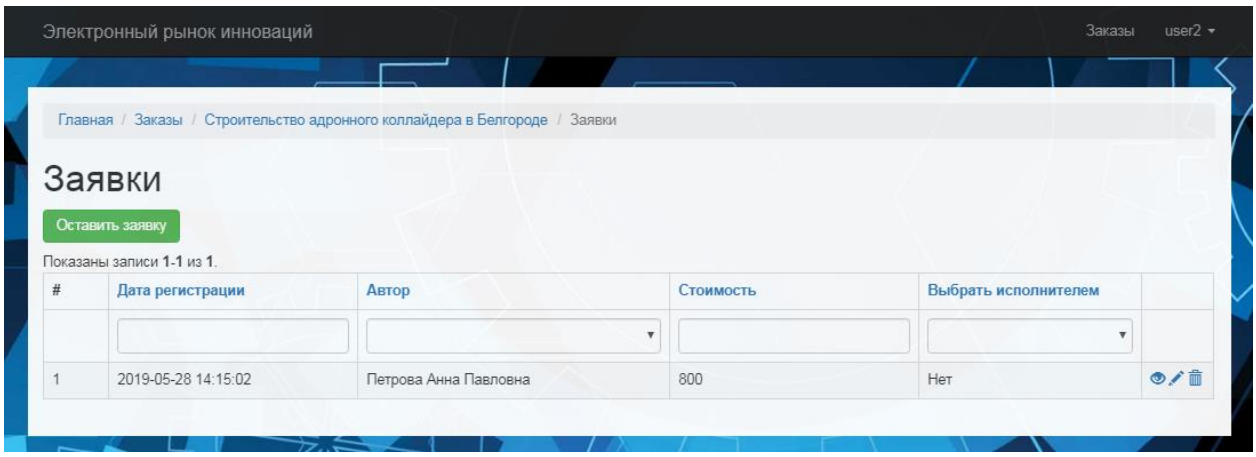

Рисунок 3.11 - Список актуальных заказов

На каждый заказ пользователь может оставить заявку.

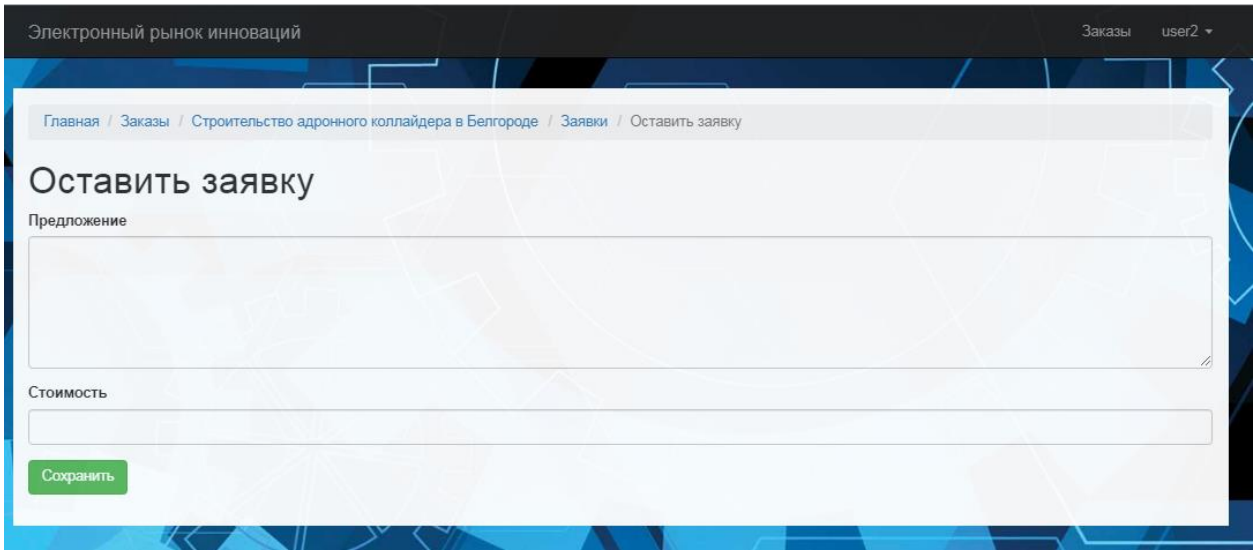

Рисунок 3.12 - Форма оставления заявки

Листинг программных модулей приведен в Приложении.

3.3 Оценка качества программы

Экспертизу качества модуля реализуют на фазах ЖЦ в рамках ГОСТ 28195-89 «Оценка качества программных средств» (ПС). Состоит она из определения номенклатуры параметров, их оценку и сравнение значений, полученных по итогу сравнения с исходными показателями. Эти параметры

качества соединены в систему из 4 уровней, описанных ниже. Допускается использовать дополнительные показатели на любом уровне.

Для поддержания доступности реализации интегральной оценки по группам показателей качества применяют факторы качества (уровень 1): стабильность ПС, настройка, удобство применения, результативность, универсальность, скорость работы.

Каждому фактору качества ставится в соответствии набор критериев качества (совокупные показатели – уровень 2): стабильность работы, работоспособность, упорядоченность, легкость конструкции, адекватность, повторяемость, простота обучения, доступность обучающей литературы, логичность эксплуатации и обслуживания, автоматизировалось, текущая эффективность, гибкость, мобильность, возможность обновления, грамотность реализации, согласованность, корректность работы функций, готовность документации, контроль доступом, резервирование, защищенность.

Критерии качества выражаются одной или несколькими метриками (уровень 3). В случае, если критерий качества выражен одной метрикой, то уровень пропускается.

Метрики включают оценочные элементы (одинарных показателей – уровень 4), отражающих указанное в метрике свойство. Общая сумма оценочных элементов, которые сводят в метрику, не ограничивается.

Параметры качества являются иерархической многоуровневой системой, где показатели высших уровней выражаются через показатели нижних уровней, и лишь на завершающем уровне оценка значений показателей реализована в рамках данных, относящихся только к ПС.

Оценка качества реализуется на всех этапах ЖЦ ПС при:

- утверждении параметров качества ПС;

- отслеживание качества на некоторых этапах разработки (ТЗ, технический и рабочий проект);

- отслеживание качества в рамках производства ПС;

- отслеживание эффективности обновления ПС в рамках сопровождения.

Оценка качества реализуют:

- разработчики в процессе создания ПС;

- фонд держатель в рамках приемки ПС в фонд;

- центры сертификации и проведения испытаний ПС в рамках внедрения;

- изготовитель в рамках копирования ПС;

- пользователь в рамках внедрения, поддержки и использования ПС.

Базовые задачи, которые решаются при оценке качества ПС:

- выделение номенклатуры уровней качества;

- выделение уровней параметров качества;

- определение методов контроля параметров качества ПС;

- отслеживание значений параметров качества;

- выполнение решений по соответствию реальных показателей качества текущим требованиям.

Способы выявления параметров качества ПС различаются:

- по методике получения данных о показателе:

- по измерению,

- по регистрации,

- по расчету,

- по восприятию человеком;

- по источникам выражения информации о ПС:

- отслеживание их функционирования в рамках работы;

- анализ заключений экспертов.

Качество программного продукта было оценено по ГОСТ 28195-89.

Данное ПО относится к подклассу 599 — прочие ПС.

В момент оценки качества ПС на каждом уровне (кроме уровня параметров оценки) реализуются вычисления показателей качества ПС, т. е. указание количественных параметров абсолютных параметров (Pij, где P – номер по порядку для i-го показателя уровня выше) и относительных параметров (Kij), которые становятся функцией показателя Pij и исходного значения Pijбаз.

Каждый параметр качества 3-го и 2-го уровней (метрика и критика) определен парой числовых параметров – значением количества и коэффициентом веса (Vij).

Сумма весовых коэффициентов показателей уровня (i) относящихся к i-му показателю вышестоящего уровня (i - 1), есть величина постоянная. Сумма весовых коэффициентов (Vij) принимается равной 1.

$$\sum_{i=1}^{n} V_j = \text{const} = 1, \qquad (1)$$

где j = 1÷n;

n - число показателей уровня (i) относящихся к i-му показателю вышестоящего уровня (i-1).

Общая оценка качества ПС формируется экспертами по набору полученных значений оценок факторов качества.

Для оценки качества ПС составляется таблица значений базовых показателей качества ПС используя метод экспертного опроса.

Определение усредненной оценки (m_{kq}) оценочного элемента по нескольким его значениям ($m_э$) проводится по формуле

$$m_{kq} = \frac{\sum_{э=1}^{t} m_э}{t} \qquad (2)$$

где t – число значений оценочного элемента (ОЭ);

k – порядковый номер метрики;

q – порядковый номер ОЭ.

Итоговая оценка k-й метрики j-го критерия ведется по формуле

$$P_{jk}^{m} = \frac{\sum_{i=1}^{Q} m_{kj}}{Q} \qquad (3)$$

где Q— число ОЭ в k-й метрике.

Абсолютные показатели критериев i-го фактора качества определяется по формуле

$$P_i = \sum_{k=1}^{n} (P_{jk}^{M} * V_{ji}^{M}) \quad (4)$$

где n - число метрик, относящихся к j-му критерию.

Относительный показатель /-го критерия i-го фактора качества вычисляется по формуле

$$K_{ij} = \frac{P_{ij}}{P_{ij}^{баз}} \quad (5)$$

Если значения базовых ПС принять идеальными, то есть равными единице, то получится следующее:

$$K_{ij} = P_{ij}^{баз} \quad (6)$$

Фактор качества ($R_i^{ф}$) вычисляется по формуле

$$R_i^{ф} = \sum_{i=1}^{N} K_i * V_{ij}^{k} \quad (7)$$

где N — число критериев качества, относящихся к i-му фактору.

Итогом показателей качества является таблица 3.2.

Таблица 3.2 - Показатели качества

Фактор	Значение показателей
Надежность	0,90
Сопровождаемость	0,76
Удобство применения	0,92
Эффективность	0,85
Универсальность	0,79
Корректность	0,78

Показатели качества ПП в целом удовлетворяют предъявляемым требованиям.

Выводы по третьему разделу

В третьем разделе ВКР была представлена схема базы данных, описана реализация системы, структура приложения и порядок его использования, а также проведена оценка качества программы.

ЗАКЛЮЧЕНИЕ

В ВКР были рассмотрены основные теоретические аспекты НИОКР, изучен документооборот, информационные потоки, специфика проведения, дано описание существующей информационной системе, а также предложены конструктивные изменения и нововведения в ней с целью организовать заказ инноваций более эффективно.

В первом разделе были рассмотрены теоретические основы НИОКР и управление ими, сформулированы функциональные требования к разрабатываемой системе, проведен анализ работы рынка инноваций НИУ БелГУ.

Во втором разделе ВКР рассмотрены основные положения проекта по разработке ИС для электронного рынка инноваций НИУ БелГУ, в том числе выбраны стратегия автоматизации и архитектура системы, представлены обоснования проектных решений по программному и техническому обеспечениям.

В третьей главе ВКР описана разработка ИС для электронного рынка инноваций НИУ БелГУ, в том числе база данных, порядок использования, оценено качество разработанной ИС.

В ходе написания ВКР приобретен опыт анализа деятельности организации, рассмотрения её информационных и материальных потоков, разработки информационной системы.

Исследованы проблемы в рамках системы управления тендерными заказами и определены способы решения этих проблем в виде решения об автоматизации рутинных процессов обработки информации.

.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Алистер, К. Современные методы описания функциональных требований к системам/ К. Алистер. – М.: Лори, 2017. – 288 с.

2. Баодин, Л. Теория и практика неопределенного программирования/ Л. Баодин. – М., Бином. Лаборатория знаний, 2017. – 416 с.

3. Бэзинс, Б. Java для начинающих. Объектно-ориентированный подход/ Б. Бэзинс, Эйми Б., Ванден Б. З. – М., Питер, 2018. – 688 с.

4. Березин, С.А. Начальный курс С и С++ / С. А. Березин, Б. А. – М.: Диалог-МИФИ, 2017. – 288 с.

5. Керниган, Б. У. Язык программирования С/ Б. У. Керниган, Д. М. Ритчи – М.: Вильямс, 2017. – 288 с.

6. Керниган, Б. У. Практика программирования/ Б. У. Керниган, Р. Пайк. – М., Вильямс, 2017. – 288 с.

7. Васильев, А.В. Самоучитель С++ с примерами и задачами / А. В. Васильев. – М.: Наука и Техника, 2015. – 480 с.

8. Васильев, Р.А. Стратегическое управление информационными системами / Р.А Васильев, Г.А. Калянов, Г.А. Левочкина. – М.: Интернет-университет информационных технологий, Бином. Лаборатория знаний, 2017. – 512 с.

9. Вернон, В. Реализация методов предметно-ориентированного проектирования / В. Вернон. – М.: Вильямс, 2017. – 688 с.

10. Габасов, Р.А. Методы линейного программирования. Часть 1. Общие задачи / Р.А. Габасов, Ф.А. Кириллова. – М.: Либроком, 2018. – 176 с.

11. Габасов, Р.А. Методы линейного программирования. Часть 3. Специальные задачи / Р.А. Габасов, Ф.А. Кириллова. – М.: Либроком, 2018. – 368 с.

12. Гамма, Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, А. Джонсон, Д. Влиссидес. –

СПб.: Питер, 2013. – 368 с.

13. Гольштейн, Е.А. Специальные направления в линейном программировании / Е.А. Гольштейн, Д.А. Юдин. – М.: Красанд, Editorial URSS, 2018. – 526 с.

14. Грацианова, Т.А. Программирование в примерах и задачах / Т.А. Грацианова. – М.: Лаборатория знаний, 2018. – 368 с.

15. Гриффитс, И. Программирование на C# 5.0 / И. Гриффитс. – М.: Эксмо, 2014. – 1135 с.

16. Данилин, А. А. Архитектура и стратегия. "Инь" и "янь" информационных технологий / А. А. Данилин, А. В. Слюсаренко. – М: Интернет-университет информационных технологий, 2017. – 506 с.

17. Дейтел, А. Как программировать на Visual C# 2012. Включая работу на Windows 7 и Windows 8 / А. Дейтел, К. Дейтел. – СПб.: Питер, 2014. – 864 с.

18. Рихтер, Д. CLR via C#. Программирование на платформе Microsoft.NET Framework 4.5 на языке C# / Д. Рихтер. – М.: Питер, 2017. – 896 с.

19. Скит, Д. C# для профессионалов. Тонкости программирования / Д. Скит. – М.: Вильямс, 2017. – 608 с.

20. Грас, Д. Data Science. Наука о данных с нуля / Д. Грас. – М.: БХВ-Петербург, 2018. – 336 с.

21. Кнут, Д. Э. Искусство программирования. Том 2. Получисленные алгоритмы / Д. Э. Кнут. – М.: Вильямс, 2017. – 832 с.

22. Кнут, Д. Э. Искусство программирования. Том 3. Сортировка и поиск / Д. Э. Кнут. – М.: Вильямс, 2017. – 824 с.

23. Дронов, В. А., Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS / В. А. Дронов. – СПб.: БХВ-Петербург, 2018. – 768 с.

24. Иванова, Г. А. Технология программирования / Г. А. Иванова. – М.: КноРус, 2018. – 336 с.

25. Исаев, Г.А. Теоретико-методологические основы качества

информационных систем / Г. А. Исаев. – М.: Инфра-М, 2018. – 258 с.

26. Исаев, Г.А. Информационные системы в экономике. Учебник / Г. А. Исаев. – М.: Омега-Л, 2013. – 462 с.

27. Исаев, Г.А. Проектирование информационных систем. Учебное пособие / Г. А. Исаев. – М.: Омега-Л, 2015. – 424 с.

28. Кузнецов, А. С. Многоэтапный анализ архитектурной надежности и синтез отказоустойчивого программного обеспечения сложных систем / А. С. Кузнецов, С. В Ченцов. – М.: Инфра-М, 2018. – 144 с.

29. Курлов, А. А. Методология информационной аналитики / А. А. Курлов, Е. А. Петров. – М.: Проспект, 2014. – 384 с.

30. Лазарева, Е. А. Основы программирования / Е. А. Лазарева. – СПб.: Питер, 2013. – 928 с.

31. Атчисон, Л. Масштабирование приложений. Выращивание сложных систем / Л. Атчисон. – М.: Питер, 2018. – 256 с.

32. Макаровских, Т.А. Языки и методы программирования. Создание простых GUI-приложений с помощью Visual C++ / Т. А. Макаровских, А.С. Панюков. – М.: Ленанд, 2018. – 144 с.

33. Фаулер, М. Предметно-ориентированные языки программирования / М. Фаулер. – М.: Вильямс, 2017. – 576 с.

34. Окулов, С. А. Динамическое программирование / С. А. Окулов, О. Г Пестов. – М.: Бином. Лаборатория знаний, 2017. – 296 с.

35. Панюкова, Т. А. Языки и методы программирования. Путеводитель по языку C++ / Т. А. Панюкова, А. С. Панюков. – М.: Ленанд, 2018. – 216 с.

36. Подбельский, В. А. Курс программирования на языке Си / В. А. Подбельский. – М.: ДМК Пресс, 2018. – 384 с.

37. Дюваль, П. М. Непрерывная интеграция. Улучшение качества программного обеспечения и снижение риска / П. М. Дюваль, С. Матиас, Э. Гловер. – М.: Вильямс, 2017. – 240 с.

38. Липпман, С. Б. Язык программирования C++. Базовый курс / С. Б. Липпман, Ж. Лажойе, Б. Э. Му. – М.: Вильямс, 2017. – 1120 с.

39. Прата, С. Язык программирования C++. Лекции и упражнения / С. Прата. – М.: Вильямс, 2017. – 1248 с.

40. Скиена, С. С. Алгоритмы. Руководство по разработке / С. С. Скиена. – СПб.: БХВ-Петербург, 2018. – 720 с.

41. Таненбаум, А. А. Современные операционные системы / А. А. Таненбаум, Е. В. Бос. – СПб.: Питер, 2015. – 1120 с.

42. Уильямс, Э. Параллельное программирование на C++ в действии. Практика разработки многопоточных программ / Э. Уильямс. – М.: ДМК-Пресс, 2014. – 672 с.

43. Брукс, Ф. Проектирование процесса проектирования. Записки компьютерного эксперта / Ф. Брукс. – М.: Вильямс, 2017. – 464 с.

44. Ховард, А. Как написать безопасный код на C++, Java, Perl, PHP, ASP.NET / А. Ховард, В. Лебланк, Д. Виега. – М.: ДМК-Пресс, 2014. – 288 с.

45. Царев, Ю. Р. Мультиверсионное программное обеспечение. Алгоритмы голосования и оценка надёжности: Монография / Ю. Р. Царев. – М.: Инфра-М, 2018. – 118 с.

46. Чистов, Д. А. Экономическая информатика (для бакалавров). Учебное пособие / Д. А. Чистов. – М.: КноРус, 2014. – 512 с.

47. Еременко М.В. ГОСУДАРСТВЕННАЯ ПОЛИТИКА В СФЕРЕ РЕГУЛИРОВАНИЯ НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ И ОПЫТНО-КОНСТРУКТОРСКИХ РАБОТ // Материалы IX Международной студенческой научной конференции «Студенческий научный форум» Режим доступа: https://scienceforum.ru/2017/article/2017032682

ПРИЛОЖЕНИЕ

Фрагмент программного кода создания заказа на исследование

```
<?php

namespace app\controllers;

use Yii;
use app\models\Tender;
use app\models\TenderSearch;
use app\models\TenderRequest;
use app\models\TenderRequestSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\base\UserException;

/**
 * Класс "TenderController" реализует действия CRUD (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») для модели класса "Tender".
 */
class TenderController extends Controller
{
    /**
     * Метод, определяющий поведение контроллера (правила доступа к действиям контроллера)
     */
        public function behaviors()
    {
        return [
                        'access' => [
                                'class' => AccessControl::className(),
                                'rules' =>
                                [
                                        [
                                                'actions' => ['index', 'view', 'create', 'update', 'delete'],
                                                'allow' => true,
                                                'roles' => ['@'],
                                                /*
                                                'matchCallback' => function ($rule, $action)
                                                {
                                                        return Yii::$app->user->identity->IsAdmin;
                                                }
                                                */

                                        ],
```

```
                                ],
                            ],

            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }


    /**
     * Действие "index" генерирует страницу вывода списка всех моделей класса "Tender".
     * @return mixed
     */
    public function actionIndex()
    {
        $searchModel = new TenderSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

                    if (!Yii::$app->user->identity->IsAdmin) {
                            $dataProvider->query->orWhere(['publish' => true])->orWhere(['id_creator'
=> Yii::$app->user->identity->id]);
                    }


        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
        ]);
    }

    /**
     * Действие "view" отображает данные конкретной модели класса "Tender".
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не
найдена
     */
    public function actionView($id)
    {
                    $model = $this->findModelForView($id);

                    $searchModel = new TenderRequestSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
                    $dataProvider->query->andWhere(['id_tender' => $model->id, 'selected' => true]);

        return $this->render('view', [
                            'searchModel' => $searchModel,
```

```
            'dataProvider' => $dataProvider,
            'model' => $model,
        ]);
    }

    /**
     * Действие "create" создает новый экземпляр класса "Tender" (новую модель).
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
     * @return mixed
     */
    public function actionCreate()
    {
        $model = new Tender();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
                            $model->registration_date = date('Y-m-d H:i:s');
                            $model->id_creator = Yii::$app->user->identity->id;
                            $model->save();

            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('create', [
            'model' => $model,
        ]);
    }

    /**
     * Действие "update" реализует редактирование выбранной модели класса "Tender".
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('update', [
            'model' => $model,
        ]);
    }

    /**
     * Действие "delete" - удаление выбранной модели класса "Tender".
```

```
    * Если операция успешно выполнена, браузер будет перенаправлен на страницу 'index'.
    * @param integer $id
    * @return mixed
    * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
    */
   public function actionDelete($id)
   {
      $model = $this->findModel($id);

                  $model->delete();

      return $this->redirect(['index']);
   }

   /**
    * Метод поиска модели класса "Tender" по первичному ключу.
    * Если модель не найдена, выдает исключение HTTP 404.
    * @param integer $id
    * @return Объект класса "Tender" (модель)
    * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
    */
   public static function findModel($id)
   {
                  $model = Tender::findOne($id);

      if ($model !== null) {
                        if (!Yii::$app->user->identity->IsAdmin) {
                              if ($model->id_creator != Yii::$app->user->identity->id) {
                                    throw new UserException (Yii::t('app', 'Access denied'));
                              }
                        }

         return $model;
      }

      throw new NotFoundHttpException(Yii::t('app', 'The requested page does not exist.'));
   }

         /**
    * Метод поиска модели класса "Tender" по первичному ключу.
    * Если модель не найдена, выдает исключение HTTP 404.
    * @param integer $id
    * @return Объект класса "Tender" (модель)
    * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
    */
   public static function findModelForView($id)
   {
                  $model = Tender::findOne($id);
```

```
        if ($model !== null) {
                        if (!Yii::$app->user->identity->IsAdmin) {
                                if (!$model->publish) {
                                        throw new UserException (Yii::t('app', 'Access denied'));
                                }
                        }
            return $model;
        }

        throw new NotFoundHttpException(Yii::t('app', 'The requested page does not exist.'));
    }
} <?php

namespace app\controllers;

use Yii;
use app\models\TenderFile;
use app\models\TenderFileSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\base\UserException;

/**
 * Класс "TenderFileController" реализует действия CRUD (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») для модели класса "TenderFile".
 */
class TenderFileController extends Controller
{
    /**
     * Метод, определяющий поведение контроллера (правила доступа к действиям контроллера)
     */
        public function behaviors()
    {
        return [
                        'access' => [
                                'class' => AccessControl::className(),
                                'rules' =>
                                [

                                        [
                                                'actions' => ['index', 'view', 'create', 'update', 'delete', 'download'],
                                                'allow' => true,
                                                'roles' => ['@'],
                                                /*
                                                'matchCallback' => function ($rule, $action)
                                                {
                                                        return Yii::$app->user->identity->IsAdmin;
                                                }
```

```
                                        */

                                            ],
                                        ],
                                    ],

            'verbs' => [
                'class' => VerbFilter::className(),
                'actions' => [
                    'delete' => ['POST'],
                ],
            ],
        ];
    }


    /**
     * Действие "index" генерирует страницу вывода списка всех моделей класса "TenderFile".
     * @return mixed
     */
    public function actionIndex($id_tender)
    {
                    $tender = TenderController::findModelForView($id_tender);

        $searchModel = new TenderFileSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
                    $dataProvider->query->andWhere(['id_tender' => $tender->id]);

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
                                'tender' => $tender,
        ]);
    }

    /**
     * Действие "view" отображает данные конкретной модели класса "TenderFile".
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не
найдена
     */
    public function actionView($id)
    {
                    $model = $this->findModel($id);
                    $tender = TenderController::findModelForView($model->id_tender);
        return $this->render('view', [
            'model' => $model,
                                'tender' => $tender,
        ]);
    }
```

```
        /**
    * Действие "download" реализует доступ к скачиванию файла.
    * @return mixed
    */
        public function actionDownload ($id)
        {
                $model = $this->findModel($id);
                $tender = TenderController::findModelForView($model->id_tender);
                $storagePath = Yii::getAlias(Yii::$app->params['ContentDirectory']);
                $filename = $model->id . '.data';
                if (!is_file("$storagePath/$filename")) {
                        throw new \yii\web\NotFoundHttpException(Yii::t('app', 'The file does not exists.'));
                }
                return Yii::$app->response->sendFile("$storagePath/$filename", $model->file_name);
        }

    /**
     * Действие "create" создает новый экземпляр класса "TenderFile" (новую модель).
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
     * @return mixed
     */
    public function actionCreate($id_tender)
    {
                $tender = TenderController::findModel($id_tender);
        $model = new TenderFile();

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
                        $model->registration_date = date('Y-m-d H:i:s');
                        $model->id_tender = $tender->id;
                        $model->save();
                        $model->UploadFile();
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('create', [
            'model' => $model,
                        'tender' => $tender,
        ]);
    }

    /**
     * Действие "update" реализует редактирование выбранной модели класса "TenderFile".
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
```

```
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
                $tender = TenderController::findModel($model->id_tender);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
                        $model->UploadFile();
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('update', [
            'model' => $model,
                        'tender' => $tender,
        ]);
    }

    /**
     * Действие "delete" - удаление выбранной модели класса "TenderFile".
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу 'index'.
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public function actionDelete($id)
    {
        $model = $this->findModel($id);
                $tender = TenderController::findModel($model->id_tender);

                $model->delete();

        return $this->redirect(['index', 'id_tender' => $tender->id]);
    }

    /**
     * Метод поиска модели класса "TenderFile" по первичному ключу.
     * Если модель не найдена, выдает исключение HTTP 404.
     * @param integer $id
     * @return Объект класса "TenderFile" (модель)
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public static function findModel($id)
    {
                $model = TenderFile::findOne($id);
        if ($model !== null) {
            return $model;
        }

        throw new NotFoundHttpException(Yii::t('app', 'The requested page does not exist.'));
    }
}
```

```
<?php

namespace app\controllers;

use Yii;
use app\models\TenderRequest;
use app\models\TenderRequestSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;
use yii\base\UserException;

/**
 * Класс "TenderRequestController" реализует действия CRUD (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») для модели класса "TenderRequest".
 */
class TenderRequestController extends Controller
{
   /**
    * Метод, определяющий поведение контроллера (правила доступа к действиям контроллера)
    */
        public function behaviors()
   {
      return [
                        'access' => [
                                'class' => AccessControl::className(),
                                'rules' =>
                                [
                                        [
                                                'actions' => ['index', 'view', 'create', 'update', 'delete', 'select', 'unselect'],
                                                'allow' => true,
                                                'roles' => ['@'],
                                                /*
                                                'matchCallback' => function ($rule, $action)
                                                {
                                                        return Yii::$app->user->identity->IsAdmin;
                                                }
                                                */

                                        ],
                                ],
                        ],

         'verbs' => [
            'class' => VerbFilter::className(),
            'actions' => [
               'delete' => ['POST'],
            ],
         ],
```

```
        ];
    }


    /**
     * Действие "index" генерирует страницу вывода списка всех моделей класса "TenderRequest".
     * @return mixed
     */
    public function actionIndex($id_tender)
    {
                $tender = TenderController::findModelForView($id_tender);

        $searchModel = new TenderRequestSearch();
        $dataProvider = $searchModel->search(Yii::$app->request->queryParams);
                $dataProvider->query->andWhere(['id_tender' => $tender->id]);

                if (!Yii::$app->user->identity->IsAdmin) {
                        if ($tender->id_creator != Yii::$app->user->identity->id) {
                                $dataProvider->query->andWhere(['id_creator' => Yii::$app->user->identity->id]);
                        }
                }

        return $this->render('index', [
            'searchModel' => $searchModel,
            'dataProvider' => $dataProvider,
                        'tender' => $tender,
        ]);
    }

            /**
     * Действие "select" определяет автора текущей заявки исполнителем.
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public function actionSelect($id)
    {

                $model = $this->findModelForView($id);
                $tender = TenderController::findModel($model->id_tender);

                $model->selected = true;
                $model->save();

        return $this->redirect(['view', 'id' => $model->id]);
    }

            /**
     * Действие "unselect" отменяет выбор.
```

```
    * @param integer $id
    * @return mixed
    * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
    */
   public function actionUnselect($id)
   {
                $model = $this->findModelForView($id);
                $tender = TenderController::findModel($model->id_tender);

                $model->selected = false;
                $model->save();

      return $this->redirect(['view', 'id' => $model->id]);
   }

   /**
    * Действие "view" отображает данные конкретной модели класса "TenderRequest".
    * @param integer $id
    * @return mixed
    * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
    */
   public function actionView($id)
   {

                $model = $this->findModelForView($id);
                $tender = TenderController::findModelForView($model->id_tender);

      return $this->render('view', [
         'model' => $model,
                        'tender' => $tender,
      ]);
   }

   /**
    * Действие "create" создает новый экземпляр класса "TenderRequest" (новую модель).
    * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
    * @return mixed
    */
   public function actionCreate($id_tender)
   {
                $tender = TenderController::findModelForView($id_tender);

      $model = new TenderRequest();

      if ($model->load(Yii::$app->request->post()) && $model->save()) {
                        $model->registration_date = date('Y-m-d H:i:s');
                        $model->id_tender = $tender->id;
                        $model->id_creator = Yii::$app->user->identity->id;
                        $model->save();
```

```
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('create', [
            'model' => $model,
                        'tender' => $tender,
        ]);
    }

    /**
     * Действие "update" реализует редактирование выбранной модели класса "TenderRequest".
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу просмотра (действие 'view').
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public function actionUpdate($id)
    {
        $model = $this->findModel($id);
                $tender = TenderController::findModelForView($model->id_tender);

        if ($model->load(Yii::$app->request->post()) && $model->save()) {
            return $this->redirect(['view', 'id' => $model->id]);
        }

        return $this->render('update', [
            'model' => $model,
                        'tender' => $tender,
        ]);
    }

    /**
     * Действие "delete" - удаление выбранной модели класса "TenderRequest".
     * Если операция успешно выполнена, браузер будет перенаправлен на страницу 'index'.
     * @param integer $id
     * @return mixed
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public function actionDelete($id)
    {

        $model = $this->findModel($id);
                $tender = TenderController::findModelForView($model->id_tender);

                $model->delete();

        return $this->redirect(['index', 'id_tender' => $tender->id]);
    }
```

```
    /**
     * Метод поиска модели класса "TenderRequest" по первичному ключу.
     * Если модель не найдена, выдает исключение HTTP 404.
     * @param integer $id
     * @return Объект класса "TenderRequest" (модель)
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public static function findModel($id)
    {
                $model = TenderRequest::findOne($id);
        if ($model !== null) {
                        if (!Yii::$app->user->identity->IsAdmin)
                        {
                                if ($model->id_creator != Yii::$app->user->identity->id) {
                                        throw new UserException (Yii::t('app', 'Access denied'));
                                }
                        }
            return $model;
        }
        throw new NotFoundHttpException(Yii::t('app', 'The requested page does not exist.'));
    }
        /**
     * Метод поиска модели класса "TenderRequest" по первичному ключу.
     * Если модель не найдена, выдает исключение HTTP 404.
     * @param integer $id
     * @return Объект класса "TenderRequest" (модель)
     * @throws Возвращает исключение класса "NotFoundHttpException", если модель не найдена
     */
    public static function findModelForView($id)
    {
                $model = TenderRequest::findOne($id);
        if ($model !== null) {
                        if (!Yii::$app->user->identity->IsAdmin)
                        {
                                $tender = TenderController::findModelForView($model->id_tender);

                                if ($model->id_creator != Yii::$app->user->identity->id && $tender->id_creator != Yii::$app->user->identity->id) {
                                        throw new UserException (Yii::t('app', 'Access denied'));
                                }
                        }
            return $model;
        }
        throw new NotFoundHttpException(Yii::t('app', 'The requested page does not exist.'));
    }
}
```