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

ИНСТИТУТ ИНЖЕНЕРНЫХ И ЦИФРОВЫХ ТЕХНОЛОГИЙ
КАФЕДРА ИНФОРМАЦИОННЫХ И РОБОТОТЕХНИЧЕСКИХ СИСТЕМ

ИНФОРМАЦИОННАЯ СИСТЕМА ПОДАЧИ И ОБРАБОТКИ ОБРАЩЕНИЙ В IT АУТСОРСИНГ

Магистерская диссертация
обучающегося по направлению подготовки
09.04.02 Информационные системы и технологии
очной формы обучения, группы 12001735
Новикова Дмитрия Михайловича

Научный руководитель
к.т.н., доцент
Титов А.И.

Рецензент
Начальник отдела эксплуатации технологической инфраструктуры филиала ФКУ «Налог-Сервис» ФНС России в Белгородской области
Мирошников Е.М

БЕЛГОРОД 2019

РЕФЕРАТ

Информационная система подачи и обработки обращений в IT аутсорсинг. – Новиков Дмитрий Михайлович, магистерская диссертация Белгород, Белгородский государственный национальный исследовательский университет (НИУ «БелГУ»), количество страниц 70, включая приложения 87, количество рисунков 46, количество таблиц 5, количество использованных источников 41.

КЛЮЧЕВЫЕ СЛОВА: информационная система, база данных, web-система, MySQL, PHP.

ОБЪЕКТ ИССЛЕДОВАНИЯ: процессы подачи и распределения обращений в IT аутсорсинг.

ПРЕДМЕТ ИССЛЕДОВАНИЯ: модели и набор алгоритмов, реализующих функционал по подаче и дальнейшей обработке обращений в IT аутсорсинг.

ЦЕЛЬ РАБОТЫ: усовершенствование методов, а так же способов подачи и обработки обращений для компаний, предоставляющих услуги IT аутсорсинга.

ЗАДАЧИ ИССЛЕДОВАНИЯ: анализ предметной области, проектирование информационной системы, проектирование базы данных, реализация алгоритмов и методов распределения заявок в системе.

МЕТОДЫ ИССЛЕДОВАНИЯ: общие принципы подачи обращений в информационных системах, процессы дальнейшей обработки данных.

ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ: в результате работы была спроектирована и реализована система подачи и обработки обращений в IT-аутсорсинг.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В магистерской диссертации создается информационная система, реализующая функционал подачи обращений в IT аутсорсинг. Данные обращения (или заявки) могут взять на исполнение зарегистрированные инженеры.

Актуальность темы магистерской диссертации можно обосновать тем, что на данный момент большое количество организаций испытывают трудности в исполнении задач в связи с высокой конкуренцией на рынке труда в части IT-специалистов. Помимо этого существуют риски, связанные с отсутствием компетенций у собственных сотрудников и необходимость их обучения. При этом организации, переходя на аутсорсинг, получают более квалифицированные услуги. Так же ведение учета выполненных работ в IT-сфере – это задача, требующая автоматизации.

Для снижения рисков и большей продуктивности, в рамках магистерской диссертации будет разработана система, позволяющая указывать нужные методы выполнения работ и распределять задачи и заявки относительно предыдущих результатов работ, которые уже были успешно достигнуты инженерами.

Магистерская диссертация состоит из трех разделов. В первом проводится анализ предметной области, анализируются уже имеющиеся системы подачи обращений, ключевые объекты предметной области и известные на данный момент способы решения поставленных задач.

Во втором разделе производится выявление объекта и предмета исследования, представлены возможные методы и алгоритмы, которые могут быть применены для достижения поставленных задач, обосновывается научная новизна. В данном разделе так же отражены разработанные функциональные модели в разрезе «как есть» и «как будет».

В третьем разделе описывается и представляется функционал разработанного предмета исследования – информационной системы подачи и обработки обращений. Описываются реализованные алгоритмы и методы, отображаются тестовые прогоны разработанной системы.

Для разрабатываемой магистерской диссертации была поставлена цель, которая заключается в усовершенствовании методов подачи и обработки обращений для компаний, предоставляющих услуги IT-аутсорсинга.

В качестве задач диссертации выделены следующие пункты:

– анализ предметной области;

– выбор методов и способов решения проанализированных проблем подачи и обработки обращений в IT аутсорсинг;

– проектирование информационной системы;

– проектирование базы данных;

– реализация алгоритмов и методов распределения заявок в системе.

1 Анализ предметной области

1.1 Предметная область

В качестве объекта исследования магистерской диссертации выступают процессы подачи и распределения обращений в IT аутсорсинг.

Предметом исследования являются модели и набор алгоритмов, реализующих функционал по подаче и дальнейшей обработке обращений в IT аутсорсинг.

Для реализации всех методов и алгоритмов информационной системы, необходим некоторый набор объектов и свойств, взаимодействующих друг с другом. В качестве выполняющих и подающих обращения лиц, выступают пользователи. Пользователь может авторизоваться в системе, описать проблему и требуемый результат решения. Аналогично, должно быть некоторое лицо, которое будет обладать некоторым набором прав, по просмотру поданных заявок со способностью взять заявку на исполнение и приступить к выполнению, установив соответствующий статус. Исполнителем заявок будет являться инженер, подходящий под указанные критерии заявки. При этом все организации обладают своим определенным спектром оказываемых услуг, ограничивающих их область деятельности. Каждая заявка будет обладать некоторым значением приоритета, влияющего на требуемую скорость реагирования выполнения заявки.

Для каждой роли пользователя разграничиваются все доступные права доступа – как к функционалу, так и к некоторым элементам интерфейса.

Таким образом, в ходе анализа, выделены следующие ключевые объекты предметной области: пользователи, права пользователей, организации, приоритеты заявок, статусы заявок, заявки и услуги организаций.

1.2 Современные методы решения

В настоящее время существует некоторый набор популярных систем подачи и учета обращений. Часть из них некоторым образом связаны с IT аутсорсингом. Большая часть систем подачи обращений основаны на принципе Help Desk систем, в которых создается некоторая заявка с набором этапов работ.

HelpDesk или так называемые системы ServiceDesk – это специализированная информационная система для решения проблем пользователей с компьютерами, аппаратным или программным обеспечением. Данные системы позволяют выявить проблемы инфраструктуры используемых информационных технологий, оценить эффективность отдела информационных технологий. Большие организации обладают развитой и сложной ИТ-инфраструктурой, функционирование всех компонентов которой является основным условием для выполнения организацией своих основных функций. Поддержка всей информационной системы в рабочем состоянии является одной из основных и главных функций ИТ-службы предприятия. Системы HelpDesk позволят ИТ-службам обеспечить оптимальное и безостановочное выполнение этой функции [1].

Системы HelpDesk обеспечивают на должном уровне:

– общую точку обращения к службе технической поддержки;

– удобные и понятные для пользователей механизмы позволяют оставлять обращения в службу поддержки, тем самым минуя менее эффективные способы решения проблем (например, попытки решить проблему самостоятельно или с помощью коллег).

– достаточно простой метод регистрации и назначения заявок специалистам;

– мониторинг очередности исполнения работ, затраченного времени на исполнения и ресурсов;

– присвоение приоритетов заявкам в зависимости от срока, пользователя или других параметров;

– мониторинг запросов и различных случаем, уведомление администраторов и менеджеров;

– учет и ведение базы знаний по выполненным запросам, позволяет ответственным лицам решать заявки, схожие с уже поступавшими;

– отчётность по затратам времени и средств на выполнение запросов.

Среди заявок, которые обслуживают системы HelpDesk, выделяют:

– заявки на обслуживание (заявки на поддержание функционирования системы);

– заявки на обработку различных случаев (под случаем понимают отклонение, которые нарушают и блокируют функционирование системы, например серьёзные сбои в системе или не выявленная и не решенная в срок неполадка, создающая серьёзные препятствия для функционирования организации);

– заявки на редактирование состояния системы – например, установку нового оборудования и программного обеспечения.

Service desk обычно рассматривается как набор из следующих компонентов:

– компонент приема заявок о проблемах и случаях;

– компонент в виде базы данных заявок;

– компонент отслеживания статусов заявок и уведомлений;

– служба с базой знаний;

– административная панель;

– компонент с модулем отчетности.

Системы HelpDesk имеют возможность интеграции со средствами учёта компьютерного оборудования. Поэтому имеется возможность осуществления общего контроля за количеством и типом оборудования, информация об

имеющемся в организации оборудовании, которое отвечает определённым требованиям (как пример, для замены вышедшего из строя).

Среди наиболее популярных HelpDesk систем в России можно выделить такие системы как:

– HappyDesk;

– HelpDeskEddy;

– vsDesk;

– Okdesk.

В качестве преимуществ системы HappyDesk можно выделить легкую настройку маршрутов коммуникации: каждому сотруднику предполагается своя зона ответственности. В данной системе предусмотрены шаблоны и заготовки ответов, а так же имеется профессиональный API для разработки любых технических решений. Кроме этого все данные можно проследить и проконтролировать на графиках или таблицах.

В системе HelpDeskEddy имеются широкие возможности для разграничения прав, реализована интеграция с IP-телефонией. Кроме этого данную технологию можно перенести на свой сервер и настроить все под себя. Система выдерживает нагрузку в 5 миллионов заявок. Проект постоянно развивается и обновляется.

Для системы vsDesk реализованы возможности подключения SMS-уведомлений, имеется возможность переназначения ответственных лиц под конкретную заявку, настроить права и роли. Кроме этого имеется система оценки полученного качества обслуживания и большое количество интеграций с другими приложениями и сервисами.

В системе Okdesk имеется мультиканальная регистрация, обработка, хранение и анализ входящих заявок от контрагентов, партнеров и производителей, ведется учет объектов обслуживания или локаций, клиентского оборудования и ПО в привязке к локациям, учет проведения разовых работ. Данная система поддерживает договоры, в том числе на

абонентские и периодические услуги. Так же в качестве преимущества можно отнести имеющееся мобильное приложение, которое работает даже offline.

Каждый из представленных инструментов обладает своими преимуществами и недостатками. У разных организаций отличные мнения и представления об идеальном обслуживании и разные возможности на старте. Каждая организация имеет свои специфические требования к решению имеющихся систем. Поэтому нет однозначного определения самой лучшей, удобной и идеальной HelpDesk системы.

Так же к одним из методов решения описанных проблем, является спектр корпоративных систем, созданных специально под нужды организации. Данные системы не выходят за пределы организации и являются узконаправленными и не широко известными.

Вывод по первой главе

В результате анализа предметной области выделены основные ключевые объекты, описаны предмет и объект исследования.

Так же описаны возможные современные методы решения поставленных задач в виде уже имеющихся подобного рода систем с описанием функциональных возможностей и основных преимуществ. Было выделено, что каждая из представленных систем обладает своими преимуществами и недостатками. Для современных методов решения поставленных проблем нет однозначного определения лучшей системы, так как каждая организация обладает отдельным мнением об идеальном обслуживании и возможностями на старте.

2 Проектирование информационной системы

2.1 Возможные методы и способы решения задач

Для решения поставленных задач магистерской диссертации и достижения выдвинутой цели рассмотрено несколько методов и алгоритмов, которые позволяют распределить работы относительно исполнителей. Выделены следующие оптимальные методы:

– метод Монте-Карло;

– метод BPM-ранжирования;

– генетические алгоритмы.

Метод Монте-Карло – это набор численных методов, которые основаны на извлечении большого объема реализаций случайного процесса, формирующийся таким образом, чтобы вероятностные характеристики совпадали с аналогичными величинами решаемой задачи [2].

Для расчета площади фигуры с помощью метода Монте-Карло, можно провести эксперимент: поместить фигуру в квадрат и случайным образом бросать точки в данный квадрат. Отсюда логично предположить – чем больше по величине площадь фигуры, тем наиболее часто будут попадать точки в данную фигуру.

Основываясь на этом, можно сделать допущение: при большом объеме количества точек, выбранных внутри квадрата случайно, процент точек, содержащихся в данной фигуре, приближенно равен отношению площади фигуры и площади квадрата.

Описанный метод нахождения приближенных площадей фигур и носит название метода Монте-Карло. Метод подходит для распределения задач относительно инженеров позволяя выбрать случайного исполнителя, который стоит выше среднего показателя эффективности [3].

Метод BPM-ранжирования в рамках диссертации состоит в расчете коэффициентов для каждого возможного случая и выбор наиболее оптимального результата с наибольшим показателем коэффициента. BPM – это концепция процессного управления организацией, которая рассматривает бизнес-процессы как особые ресурсы предприятия, непрерывно адаптируемые к постоянным изменениям, и полагающаяся на понятность и видимость бизнес-процессов в организации. BPM отвечает на вопросы: где, когда, зачем, как и какая работа находится на исполнении и какое ответственное лицо отвечает за выполнение [4].

Данный метод подходит для выбора наиболее оптимального исполнителя, позволяя выбрать только тех инженеров, которые выполняли данный вид заявок, рассчитать коэффициенты полезности каждого инженера по некоторым критериям и выбрать одного с наиболее высоким показателем коэффициента.

В качестве дополнительного метода для распределения заявок возможно использование генетических алгоритмов. Однако нужно учитывать, что график скорости выполнения заявки не верно трактовать как квалификацию инженера (в виду того что некоторые инженеры могут хитрить и не закончив работу по заявке перевести в статус закрытия, для уменьшения показателя срока выполнения. Тем самым завышая показатели КПД). Исходя из этого, данный тип алгоритмов не всегда является оптимальным для распределения задач.

Исходя из представленных выбранных методов и алгоритмов, сформирована научная новизна создаваемой системы:

– реализация распределения задач относительно предыдущих результатов работ, которые уже были успешно достигнуты инженерами;

– использование для распределения заявок методов «Монте-Карло» или BPM-ранжирования для реализации наиболее оптимального выбора исполнителей;

– проведение сравнительного анализа методов Монте-Карло и BPM-ранжирования.

2.2 Функциональные модели информационной системы

Для более полного и детального понимания функционирования системы, разработаны функциональные модели.

Для разработки функциональных моделей использовалось программное средство от фирмы Computer Associates седьмой версии: AllFusion Process Modeler.

AllFusion Process Modeler – это программное средство для проведения моделирования, анализа, оптимизации и документирования бизнес-процессов. AllFusion Process Modeler так же можно использовать для графического представления бизнес-процессов [5].

AllFusion Process Modeler позволяет выделять важные компоненты любых бизнес-процессов: манипуляции, которые необходимые произвести, способы осуществления и контроля, требуемые ресурсы и отразить получаемые результаты.

Построение функциональных моделей производилось в два этапа. Первый этап, отображает функциональная модель в виде «AS-IS» или «как есть». Данная модель отображает процессы того, как на данный момент функционируют подобного рода HelpDesk системы.

На втором этапе функциональная модель отображается в виде «AS-BE» или «как будет». Модель на данном этапе отображает получаемые улучшения в ходе модернизации старых принципов работы систем.

2.2.1 Функциональная модель в виде «AS-IS».

На рисунке 2.1 представлена контекстная диаграмма, созданная в нотации IDEF0. На диаграмме виден общий процесс «система подачи и обработки обращений в it аутсорсинг».

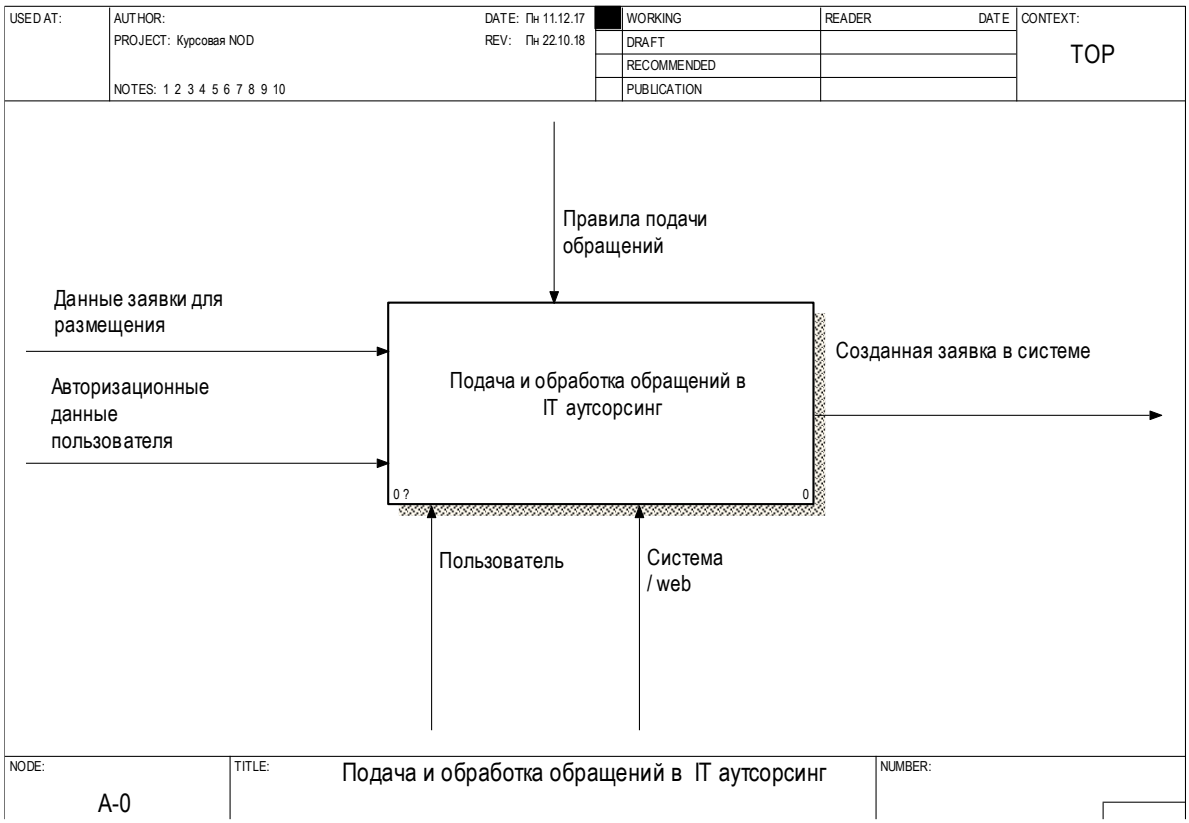


Рисунок 2.1 – Контекстная диаграмма.

На данном рисунке отображена общая схема подачи и обработки обращений в it аутсорсинг. При обращении к системе пользователь предоставляет системе данные заявки для размещения и вводит свои авторизационные данные, после чего заявка добавляется и появляется в системе. При взаимодействии с системой используются правила подачи обращений, которые указаны в качестве управляющего потока на диаграмме. Основными механизмами выполнения являются пользователь, являющийся субъектом обращения и web-система, которая обрабатывает направляемые запросы.

На рисунке 2.2 представлена декомпозиция контекстной диаграммы. Данная диаграмма является более подробным описанием действий, проводимых внутри контекстной диаграммы.

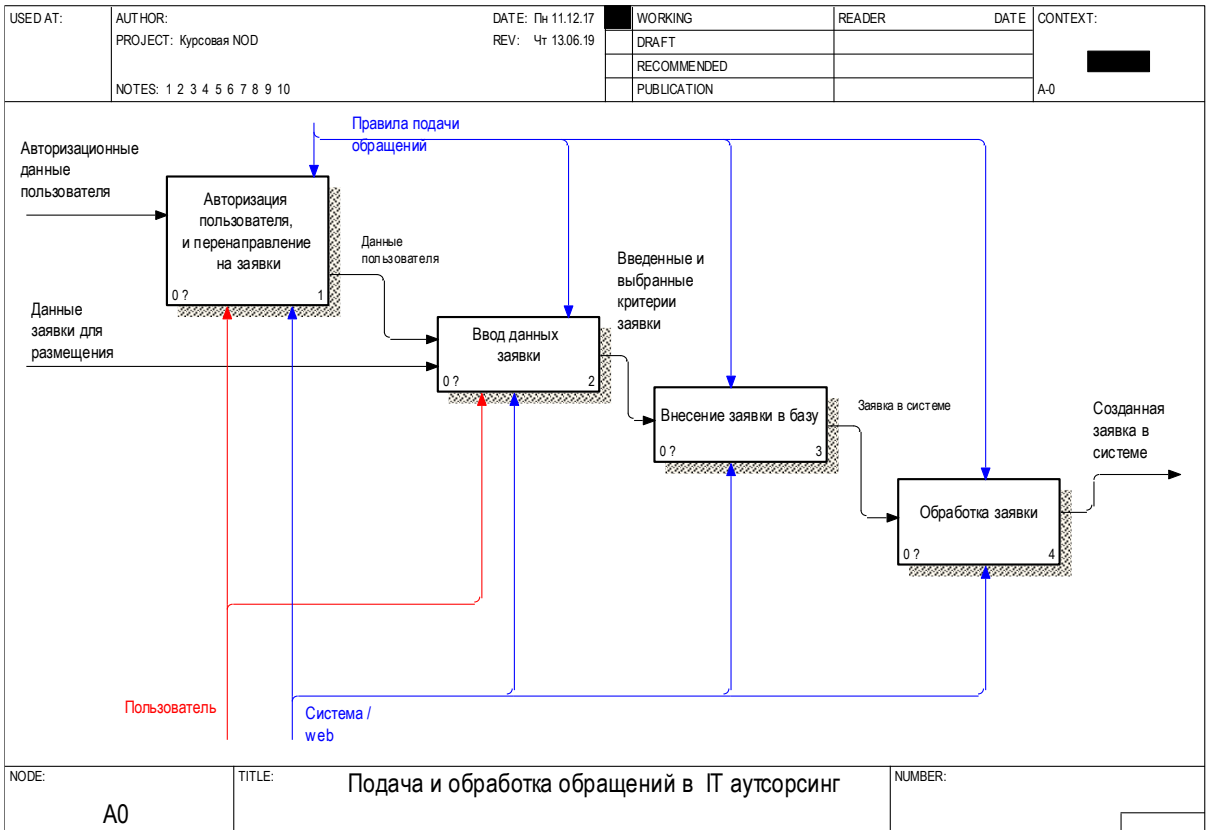


Рисунок 2.2 – Декомпозиция контекстной диаграммы.

Данная диаграмма отображает более подробную последовательность действий при подаче обращений в it аутсорсинг. Для того чтобы подать обращение, необходимо пройти следующие 4 этапа: авторизация пользователя с дальнейшим перенаправлением на список заявок, ввод данных для создания заявки, внесение заявки в базу и обработка заявки.

На рисунке 2.3 более подробно представлен процесс авторизации пользователя в виде декомпозиции данной сущности.

Как видно из рисунка 2.3, на первом этапе пользователь вводит данные для авторизации. После этого производится формирование запроса на поиск введенных данных в базе. Если в процессе выполнения запроса такой учетной записи не найдено в базе, формируется и выводится ошибка. Иначе производится вход пользователя под учетной записью.

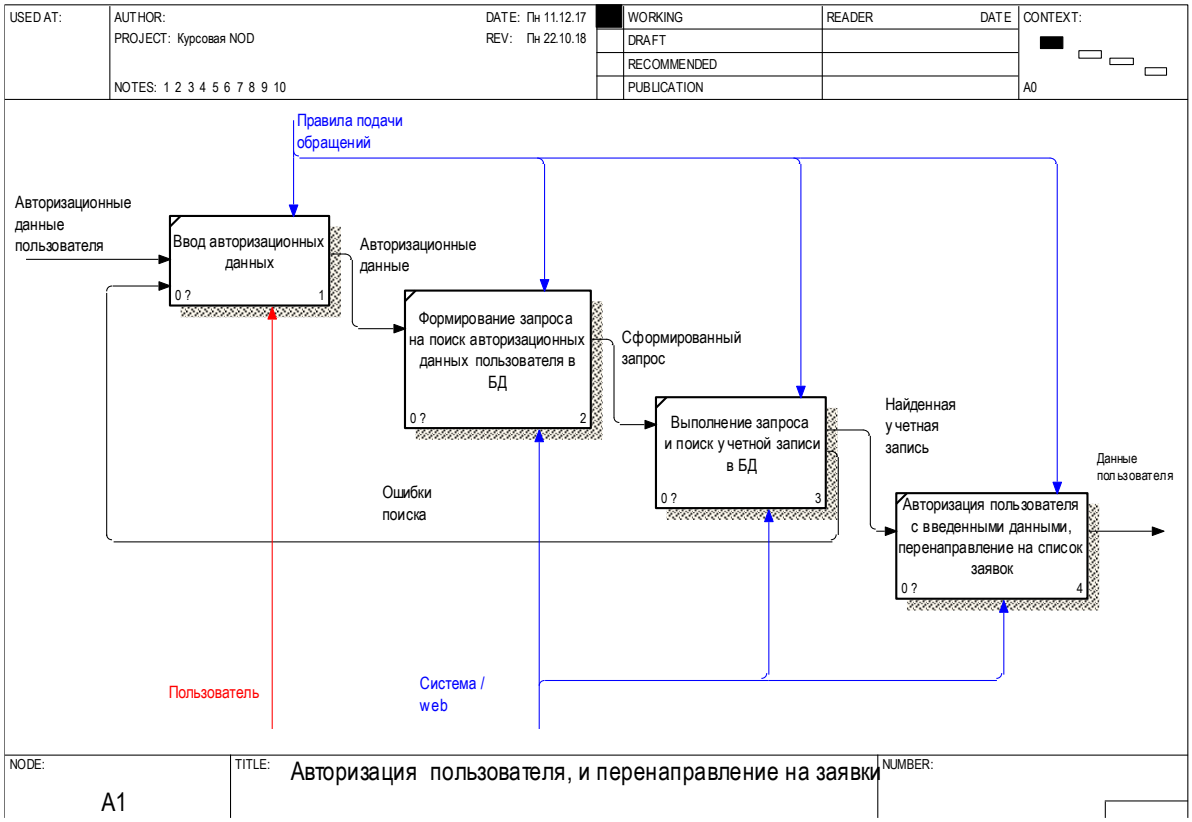


Рисунок 2.3 – Декомпозиция процесса «авторизация пользователя»

Как видно из рисунка 2.3 процессы авторизации частично исполняют как пользователь, так и разрабатываемая система, а управляющим потоком являются правила подачи обращений.

На рисунке 2.4 показан процесс ввода данных заявки. На первом этапе проверяются права на добавление заявки. На основании данных прав система определяет доступные поля для добавления или редактирования в заявках, а так же разграничивает возможности по созданию заявок.

Выходной поток в данном процессе является управляющим для процессов подгрузки организации из БД и статуса, выбора приоритета заявки, а так же для ввода описания заявки.

Проверив права пользователя, производится процесс поиска организации из базы данных, статуса и приоритета заявки. Кроме этого от пользователя требуется ввести описание заявки, после чего формируется массив введенных и выбранных данных. Данный массив является выходным потоком и используется для дальнейших обработок системой.

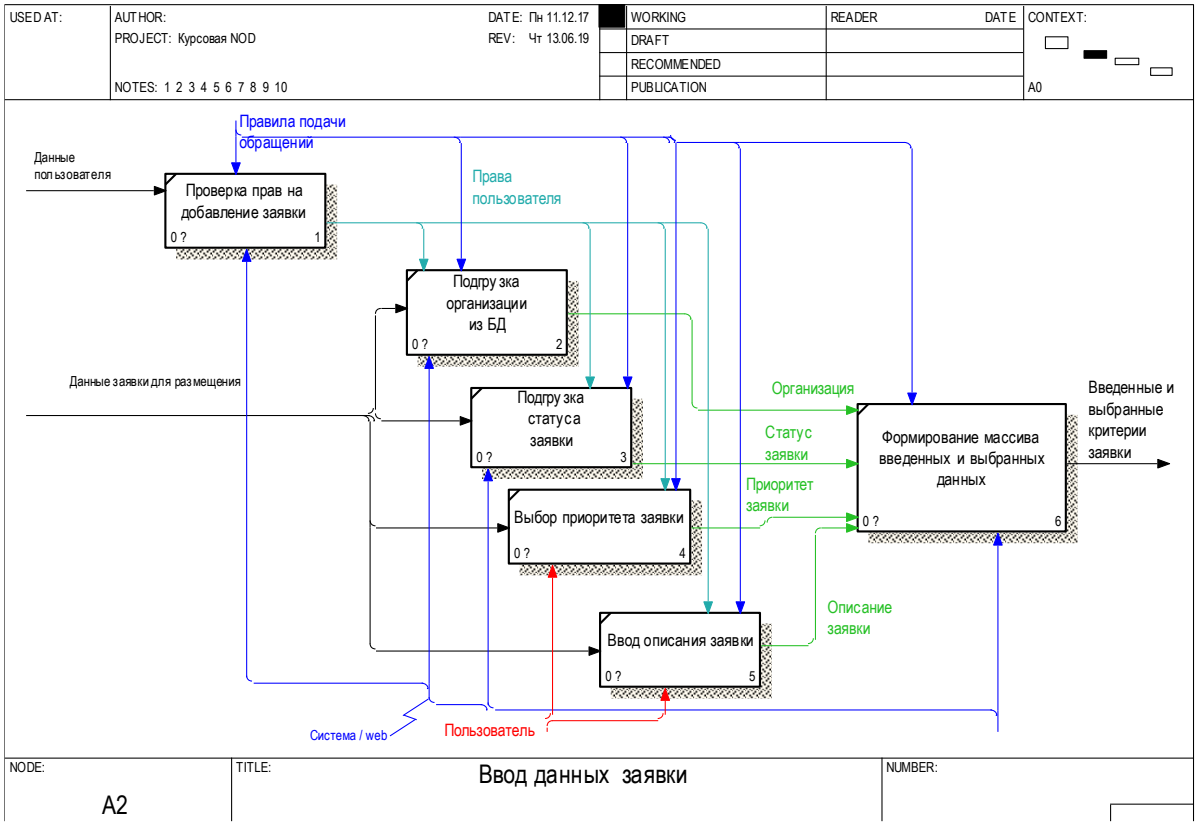


Рисунок 2.4 – Декомпозиция процесса «ввод данных заявки»

Как видно из рисунка 2.4 процессы ввода данных исполняет пользователь, а система отвечает за загрузку некоторых данных и формирование массива выбранных и введенных параметров.

Введенные и выбранные критерии заявки в дальнейшем вносятся в базу. Данный процесс представлен в декомпозиции на рисунке 2.5.

Как видно из рисунка 2.5 на первом этапе формируется массив данных заявки. На основании сформированного массива генерируется запрос к базе данных на добавление заявки. Сгенерировав, проверив и успешно выполнив запрос, запись появляется в базе данных заявок.

В случае ошибки при проверке – производится уведомление о некорректном запросе на добавление с разъяснением ошибки.

Всем процессом внесения заявок управляет создаваемая система, основываясь на правилах подачи обращений.

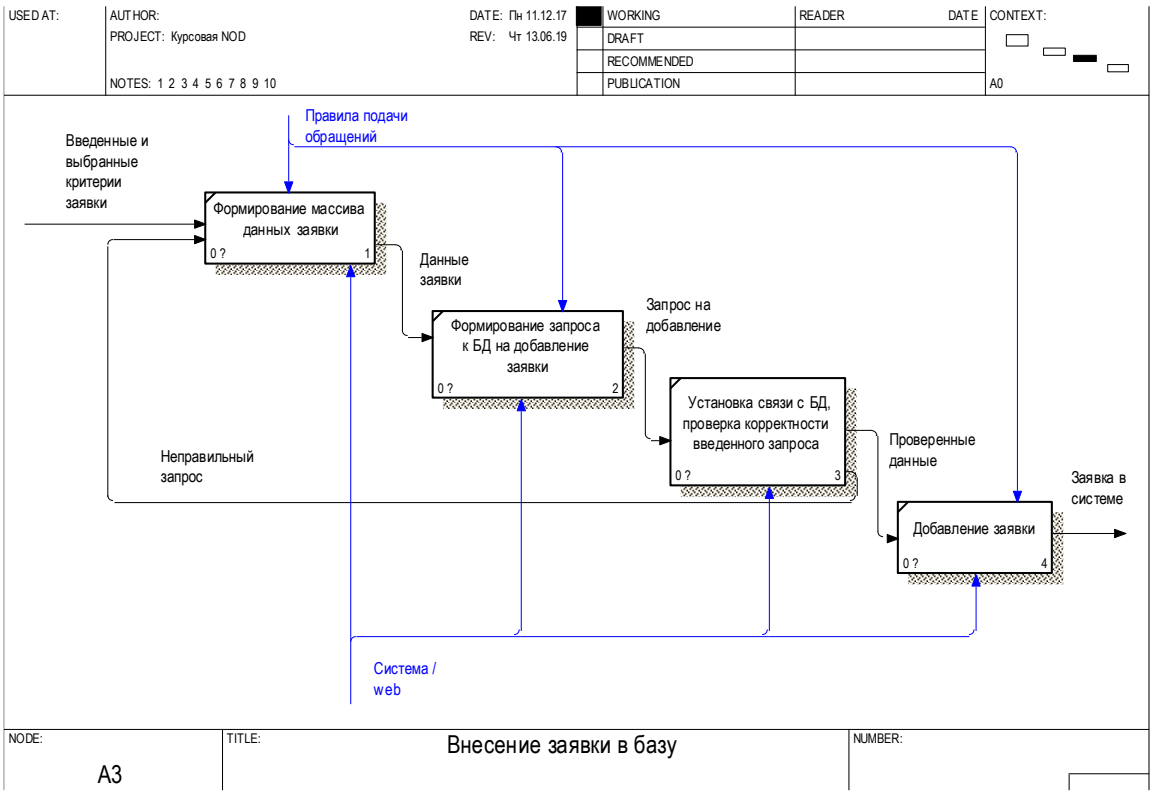


Рисунок 2.5 – Декомпозиция процесса «внесение заявки в базу»

На рисунке 2.6 показана декомпозиция процесса обработки заявки. В данном процессе происходит расчет и установка дополнительных параметров.

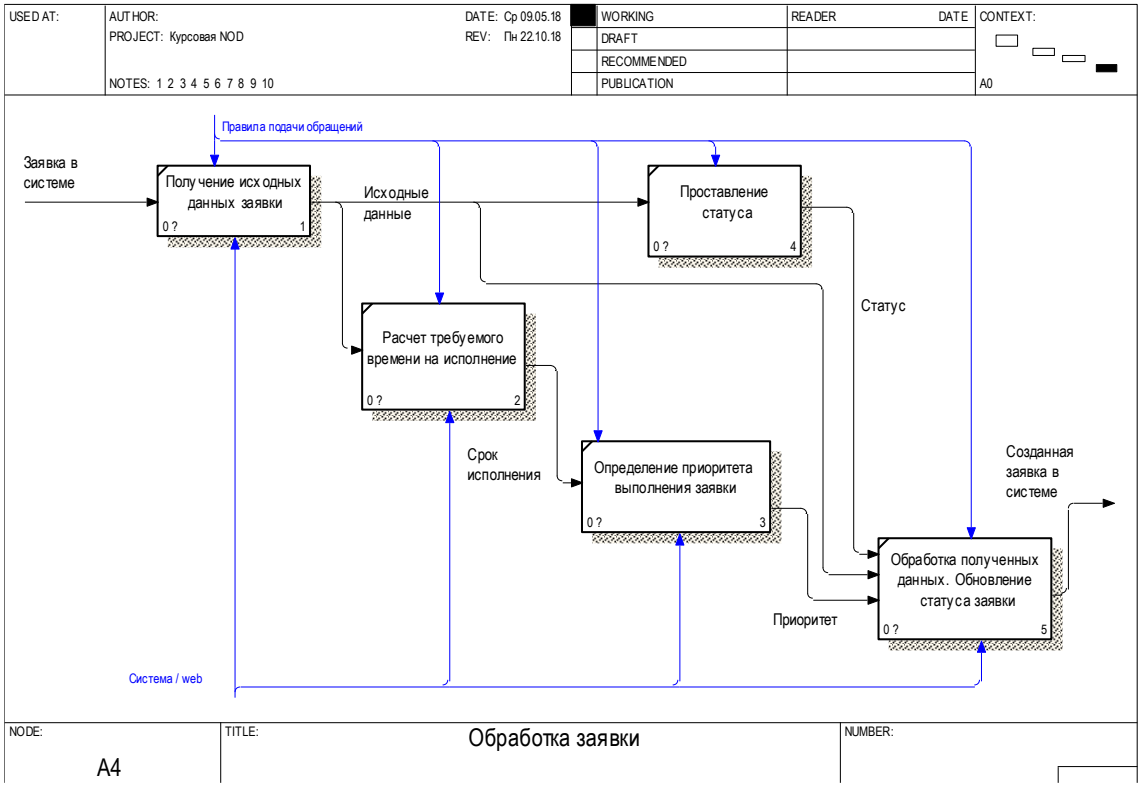


Рисунок 2.6 – Декомпозиция процесса «обработка заявки»

Как видно из рисунка 2.6 на первом шаге производится получение исходных данных заявки, далее рассчитывается требуемое время на выполнение, определяется приоритет заявки и проставляется статус. После этого происходит обновления обработанных данных заявки. На этом процесс обработки завершается.

2.2.2 Функциональная модель в виде «AS-BE».

Функциональная модель «AS-BE» претерпела некоторые изменения по сравнению с моделью «AS-IS». Все измененные и отличающиеся данные имеют зеленое цветовое выделение.

На рисунке 2.7 представлена полученная контекстная диаграмма. Как видно из рисунка, на данном этапе добавились правила работы информационных систем.

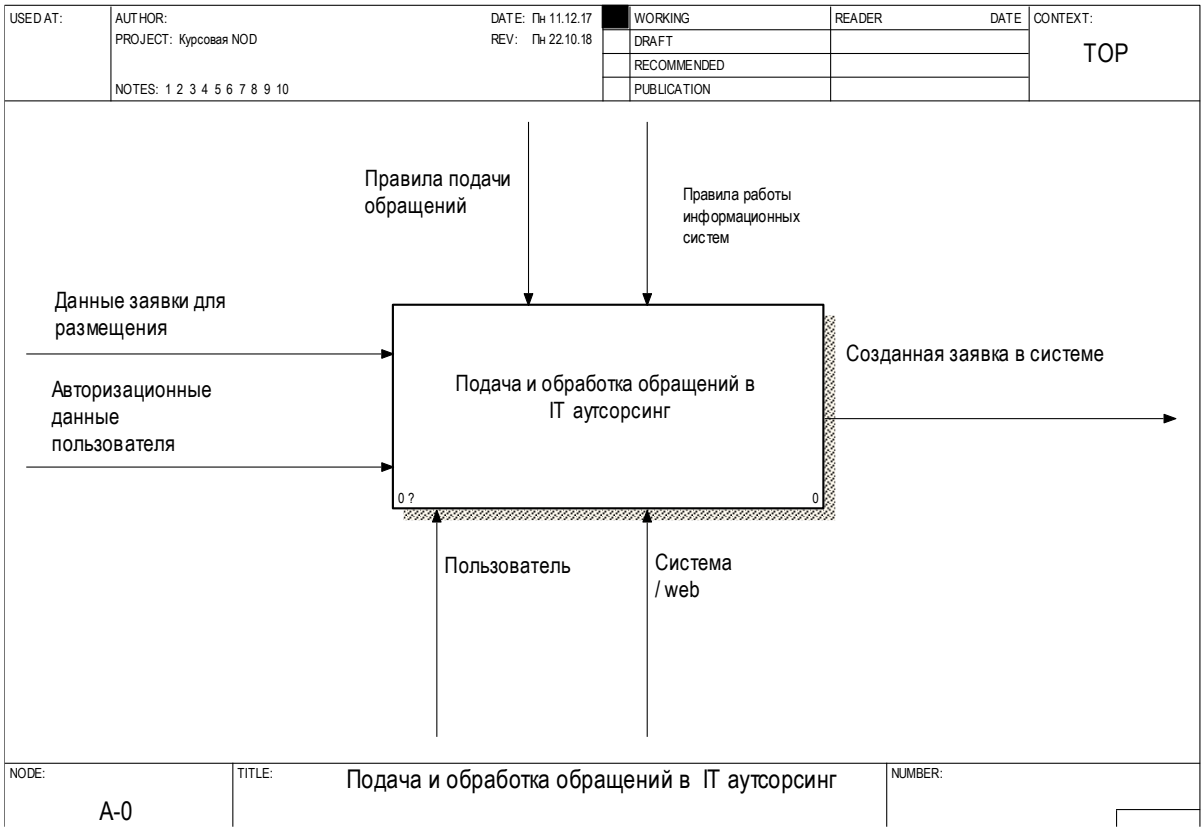


Рисунок 2.7 – Контекстная диаграмма.

На рисунке 2.8 представлена декомпозиция контекстной диаграммы. Как видно из представленной диаграммы по сравнению с моделью «как есть» добавился новый поток управления с правилами работы информационных систем. Так же появился новый процесс анализа заявки с выходным потоком в виде проанализированных данных.

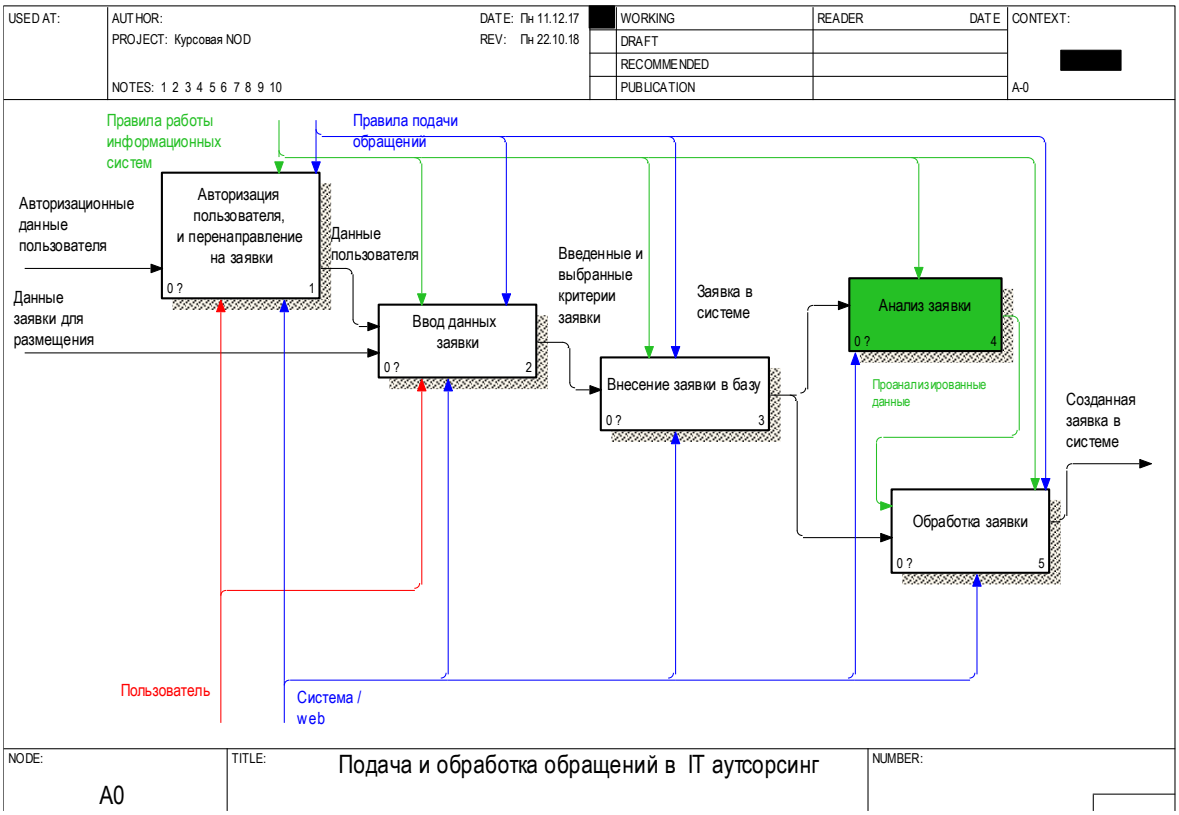


Рисунок 2.8 – Декомпозиция контекстной диаграммы.

Как видно из рисунка 2.8 после анализа данных проанализированные данные передаются входным потоком в процесс обработки заявки для дальнейших расчетов полученных данных.

На рисунке 2.9 представлен процесс авторизации пользователя в виде декомпозиции данной сущности.

Как видно из рисунка 2.9 процесс авторизации почти не поменялся. В качестве основных изменений является только управляющий поток правил работы информационных систем.

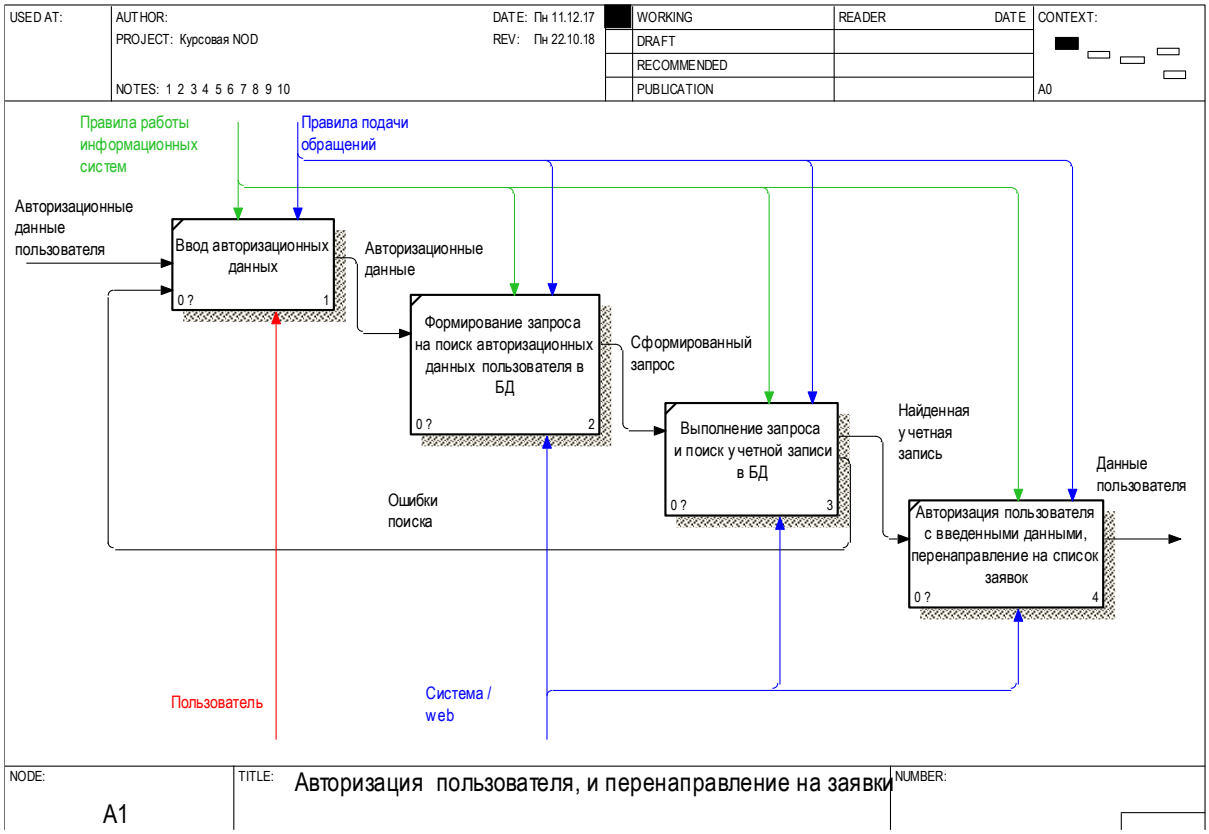


Рисунок 2.9 – Декомпозиция процесса «авторизация пользователя»

На рисунке 2.10 показан процесс ввода данных заявки на основании декомпозиции одноименного процесса. Основными изменениями на данной диаграмме являются добавленные управляющие потоки в виде правил работы информационных систем.

Так же изменился механизм выбора приоритета заявки. Теперь механизмом исполнения данного процесса является только разрабатываемая web система.

В случае возникновения ошибок при поиске учетной записи в базе данных, система так же уведомляет об ошибке и требует повторного ввода данных. В случае успешной операции поиска учетной записи – производится авторизация и перенаправление на список заявок.

Данный процесс претерпел незначительные изменения при сравнении с процессом авторизации модели «как есть».

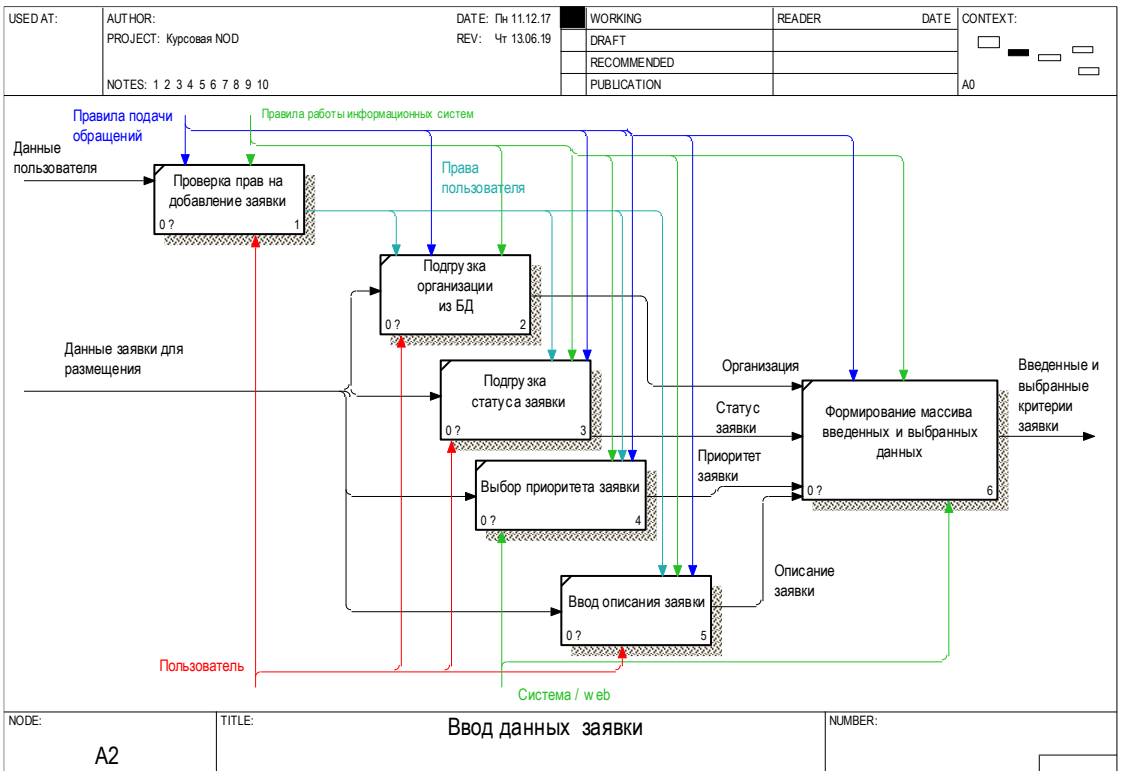


Рисунок 2.10 – Декомпозиция процесса «ввод данных заявки»

Процесс внесения заявки в базу почти не претерпел изменений. Данный процесс представлен на декомпозиции, представленной на рисунке 2.11.

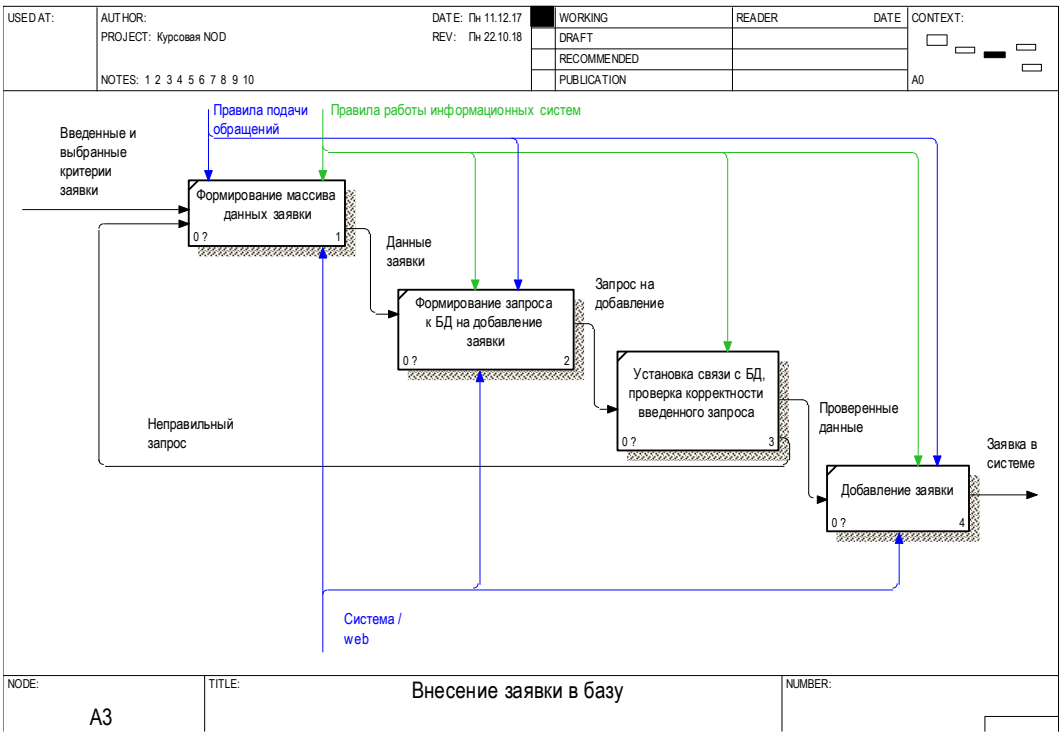


Рисунок 2.11 – Декомпозиция процесса «внесение заявки в базу»

Как видно из рисунка 2.11 основным изменением является добавление управляющего потока с правилами работы информационных систем.

На рисунке 2.12 показана декомпозиция добавленного в модели вида «как будет» процесса анализа заявки.

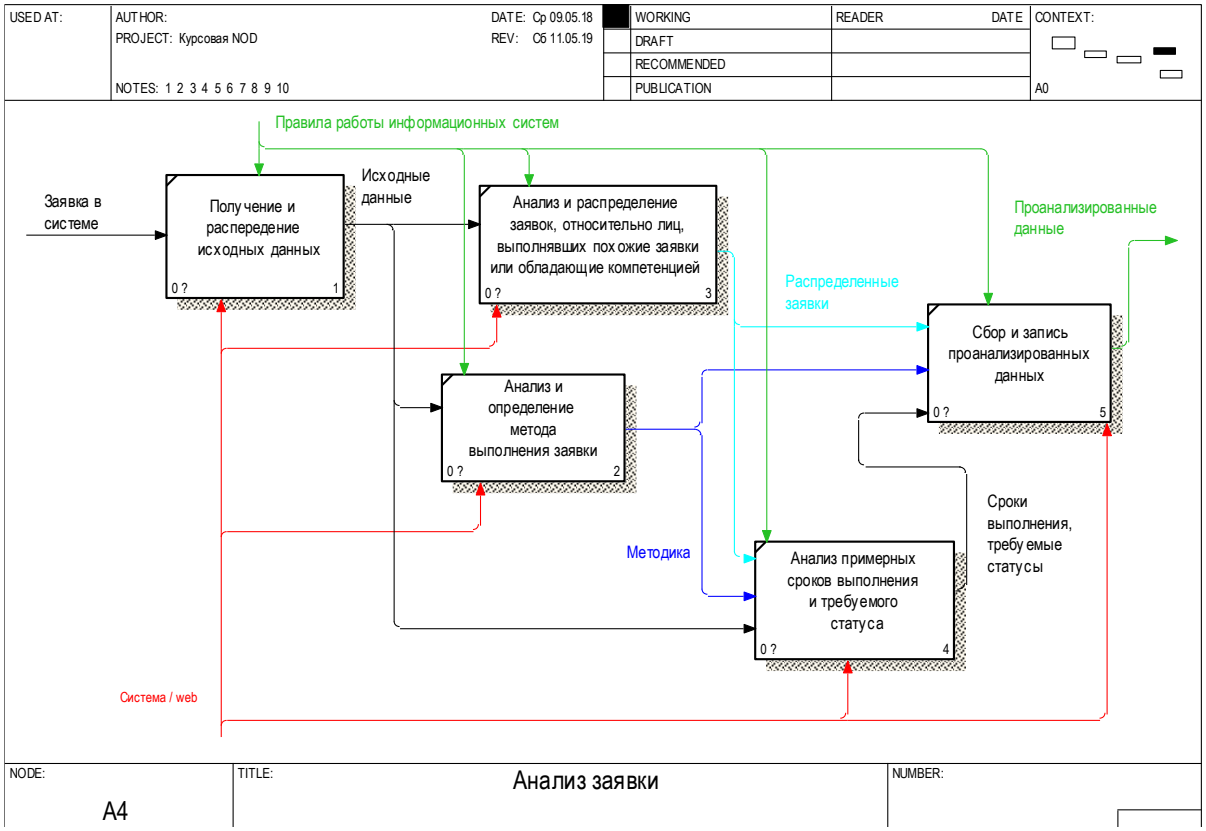


Рисунок 2.12 – Декомпозиция процесса «анализ заявки»

Как видно из рисунка 2.12 на первом этапе производится получение и распределение исходных данных заявки, далее производится анализ и распределение заявок относительно лиц, выполнявших подобные заявки. Производится анализ и определение метода выполнения заявки, примерных сроков выполнения и требуемого статуса. После этого происходит сбор и запись проанализированных данных. На этом процесс анализа завершается.

На рисунке 2.13 отображены изменения декомпозиции обработки заявки. В данном процессе добавлены входные данные в виде проанализированных данных, полученных на прошлом этапе и поток управления, описанный ранее.

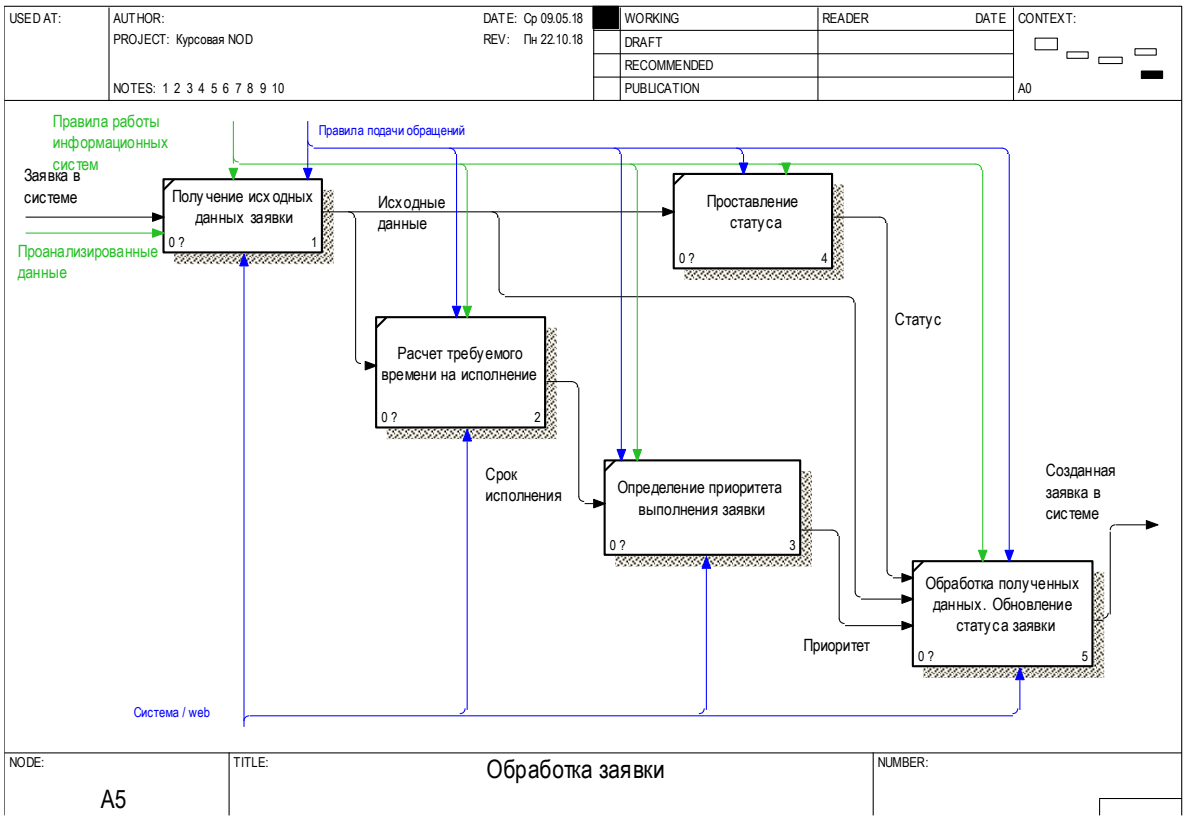


Рисунок 2.13 – Изменения декомпозиции процесса «обработка заявки»

Как видно из рисунка 2.13 основной процесс обработки заявки при сравнении с процессом из модели «как есть» почти не изменился.

Таким образом, произведено описание функциональной модели работы создаваемой системы, построены модели «как есть» и «как будет». Данное описание позволяет упростить понимание основных структур и принципов функционирования разрабатываемой системы, сделав данные процессы более прозрачными и наглядными.

2.3 Выбор инструментальных средств для реализации информационной системы

Для реализации информационной системы по подаче и обработки обращений в качестве языка программирования выбран язык PHP.

PHP – это скриптовый язык общего назначения, интенсивно применяемый для разработки веб-приложений и поддерживаемый большим

спектром хостинг-провайдеров, являясь одним из основных и популярных языков, применимых для создания динамических веб-сайтов [6].

Язык PHP и его интерпретатор разработан энтузиастами в рамках проекта с открытым кодом. При этом распространяется проект под собственной лицензией.

Синтаксис PHP отчасти похож на синтаксис языка Си. Некоторые элементы, например ассоциативные массивы и цикл foreach, заимствованы из Perl. При этом любой скрипт может начинаться непосредственно с оператора PHP [7].

Сторонники языка PHP в качестве основных его преимуществ приводят широкую распространённость, большую востребованность PHP-разработчиков на рынке труда и относительную простоту изучения. К достоинствам языка можно так же отнести и достаточно быструю эволюцию. В то же время язык иногда критикуют за несогласованный синтаксис функций и неортогональность дизайна.

В качестве среды программирования на представленном языке выбрано программное средство PhpStorm.

PhpStorm – это кросс-платформенная среда для разработки на языке PHP. Данный программной продукт разрабатывается крупной компанией JetBrains на основании популярной платформы IntelliJ IDEA [8].

Программный продукт PhpStorm представлен как интеллектуальный редактор для таких форматов как PHP, HTML и JavaScript с возможностями анализа и нахождения ошибок кода. PhpStorm поддерживает почти весь круг спецификаций PHP, включая генераторы, сопрограммы, пространства имен, замыкания, типажи и синтаксис коротких массивов. Кроме этого имеются возможности автодополнения кода. Имеется так же полнофункциональный SQL-редактор с возможностью редактирования полученных результатов запросов.

PhpStorm разработан на основе платформы IntelliJ IDEA, которая написана на Java. Пользователи имеют возможность расширять

функциональность среды разработки за счет установки или написания своих собственных плагинов [9].

2.4 Проектирование базы данных

2.4.1 Выбор СУБД

В качестве СУБД магистерской диссертации выбрана СУБД MySQL. MySQL – это реляционная система управления базами данных. Данные в базах системы хранятся в виде связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL [10]. MySQL – это свободно распространяемая система, без оплаты за применение, что является большим плюсом. Кроме этого, это достаточно быстрая, надежная и простая в использовании СУБД, которая подходит для не очень больших и глобальных проектов [11].

В MySQL работа реализована как в текстовом режиме так и в графическом. Наиболее популярным визуальным интерфейсом (написанном на PHP) для работы с этой СУБД является PhpMyAdmin. Этот интерфейс позволяет сильно упростить и наиболее удобно визуализировать работу с базами данных в СУБД MySQL [12].

PhpMyAdmin позволяет использовать все достоинства браузеров, прокрутку изображений, если они не умещаются на экран. Большое количество базовых SQL функций в PhpMyAdmin приведены к понятным интерфейсам и действиям, похожих на переход по ссылкам в Internet [13].

2.4.2 Проектирование структуры БД

Для данного этапа проектирования разработана модель в CASE-системе. Данная модель делится на два уровня: логический и физический. Логический уровень модели основан на прямом отображении фактов из реальной жизни.

Факты обладают названиями на простом естественном языке, включая любые разделители слов. Использование определенной СУБД и типов данных на логическом уровне модели не рассматриваются [14]. Вид логической модели рассматриваемой предметной области представлен на рисунке 2.14. Логический уровень – это абстрактный взгляд на данные, с представлением данных так, как они выглядят в реальном мире. Объекты модели, представляемые на логическом уровне, называются сущностями и атрибутами. Логическая модель данных является универсальной и никаким образом не связана с конкретной реализацией СУБД.

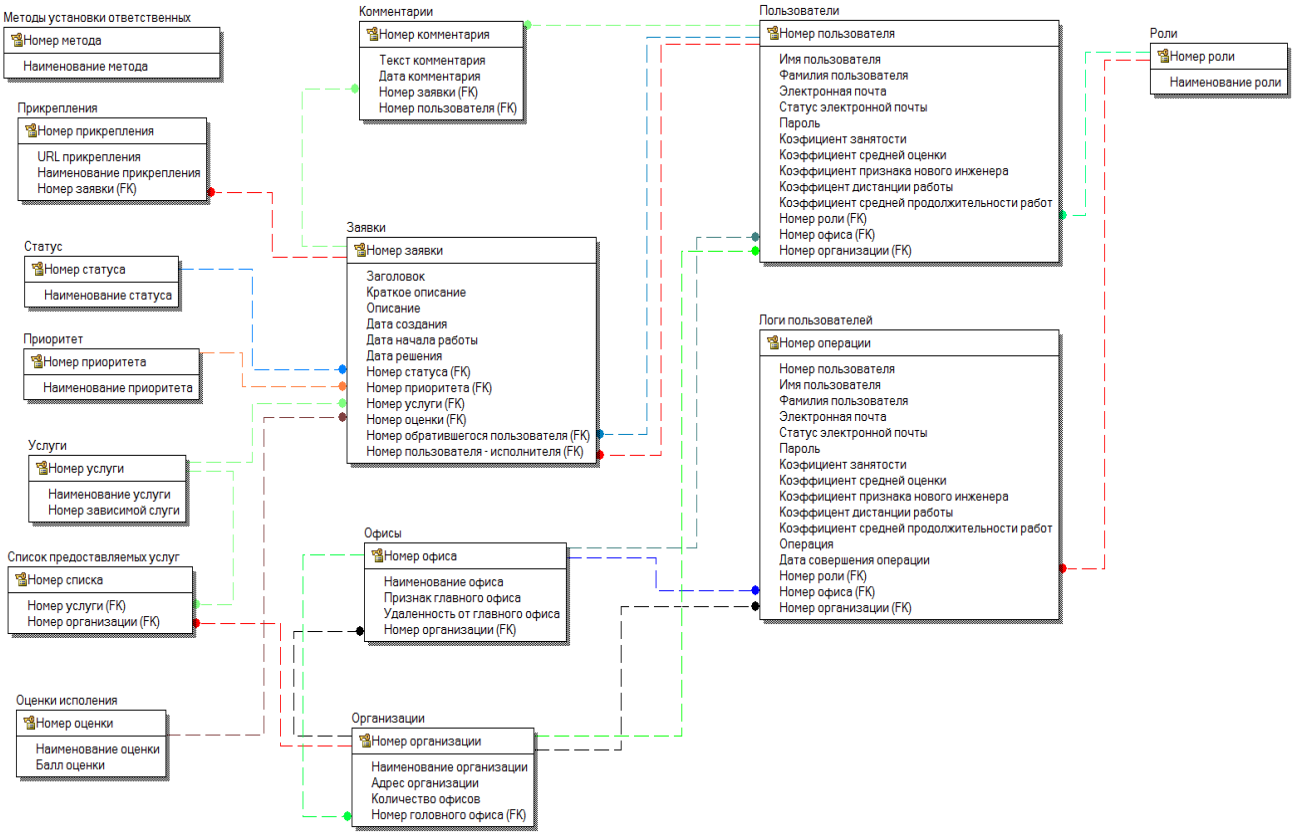


Рисунок 2.14 – Вид логической модели

Как видно из рисунка 2.14 модель обладает следующими 14 сущностями:

– методы установки ответственных отвечают за методы распределения заявок относительно инженеров;

– прикрепления отвечают за хранение данных о прикрепляемых файлах;

– статусы содержат список промежуточных этапов выполнения заявок;

– приоритеты определяют список критериев оценки важности заявок;

– услуги содержат иерархический список всех возможных услуг организаций;

– список предоставляемых услуг содержит зависимость организаций от предоставляемых ими услуг;

– оценки исполнения включают список возможных критериев оценки исполнения заявки инженером;

– комментарии хранят тексты, реализующие способ общения между исполнителем и обратившимся лицом;

– сущность заявок является основной сущностью базы и содержит в себе все параметры для каждой создаваемой заявки;

– офисы содержат данные об офисах каждой организации;

– организации включают список всех фирм, являющихся исполнителями или субъектами обращения;

– пользователи хранят учетные данные для входа в информационную систему;

– в сущности ролей содержатся определения для каждого вида прав учетных записей.

Все перечисленные сущности обладают связями, реализующими функциональное взаимодействие между своими элементами.

Второй физический уровень модели включает в себя имена объектов и типы данных. При этом физическая модель данных в отличие от логической модели, зависит от конкретной СУБД, являясь фактическим отображением системного каталога. В физической модели содержится информация всех объектов БД [15]. Так как стандартов на объекты базы данных не существует, то физическая модель зависит от реализации СУБД. Если в логической модели не имеет значения, какой тип данных имеет атрибут, то в физической модели важно описать всю информацию о конкретных физических объектах – таблицах, типах данных, индексах и прочем. Вид физической модели рассматриваемой предметной области представлен на рисунке 2.15.

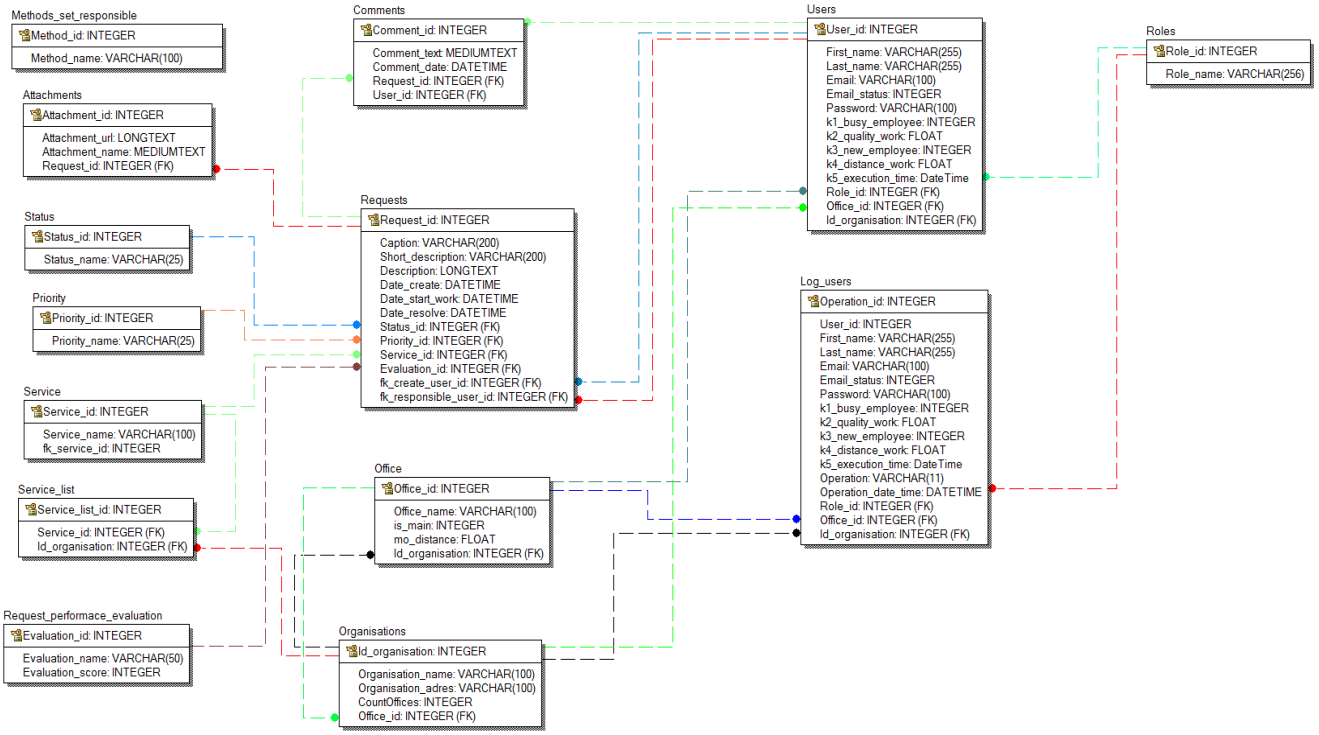


Рисунок 2.15 – Вид физической модели

Данная модель обладает таким же набором сущностей и атрибутов, которые были описаны для логической модели.

Основными таблицами базы данных являются таблицы requests, users, office и organisations.

Для таблицы request спроектированы следующие поля:

– request_id – уникальный идентификатор и номер обращения;

– caption – заголовок обращения, наиболее кратко отражающий основную суть проблемы;

– short_description – краткое описание проблемы, содержащее минимальное количество информации, которая позволит инженеру наиболее быстро понять суть проблемы;

– description – полное описание проблемы, содержащее наиболее полный и развернутый список проблем и суть задачи;

– date_create – дата создания задачи;

– date_start_work – дата начала работы над заявками, отражающая момент взятия заявки в работу ответственным инженером;

– date_resolve – дата решения заявки;

– status_id – идентификатор статуса, отражающий этап работы над заявкой и являющийся вторичным ключом, зависимым от таблицы статусов;

– priority_id – идентификатор приоритета, обозначающий требуемое время реагирования на создаваемую заявку и являющийся вторичным ключом, зависимым от таблицы приоритетов;

– service_id – идентификатор типа проблемы, которую требуется решить ответственному инженеру;

– evaluation_id – оценка выполненной заявки;

– fk_create_user_id – поле с вторичным ключом, отражающее идентификатор пользователя, создавшего заявку;

– fk_responsible_user_id – поле с вторичным ключом, отражающее идентификатор инженера, ответственного за выполнения заявки.

Для таблицы users спроектированы следующие поля:

– user_id – уникальный идентификатор пользователя;

– first_name – имя пользователя;

– last_name – фамилия пользователя;

– email – электронная почта пользователя;

– email_status – статус электронной почты или статус учетной записи, позволяющий или запрещающий пользователю авторизоваться в системе;

– password – пароль пользователя;

– k1_busy_employee – признак занятости работника, означающий свободен ли сейчас инженер для выполнения заявки;

– k2_quality_work – средняя оценка выполненных заявок инженером;

– k3_new_employee – признак нового сотрудника;

– k4_distance_work – удаленность от места выполнения текущей заявки;

– k5_execution_time – среднее время исполнения заявок;

– role_id – вторичный ключ, обозначающий идентификатор роли для учетной записи и ограничивающий доступный функционал;

– office_id – вторичный ключ, отражающий идентификатор принадлежности пользователя к офису.

– id_organisation – вторичный ключ, обозначающий принадлежность пользователя к организации.

Спроектированные поля для таблицы office имеют следующую структуру:

– office_id – уникальный идентификатор офиса;

– office_name – наименование офиса;

– is_main – признак того, что офис является головным;

– mo_distance – удаленность от головного офиса;

– id_organisation – идентификатор принадлежности офиса к организации.

Для таблицы organisations спроектированы следующие поля:

– id_organisation – уникальный идентификатор организации;

– organisation_name – название организации;

– organisation_adres – адрес организации;

– countOffices – количество офисов организации;

– office_id – вторичный ключ, обозначающий принадлежность к офису.

На этом процесс проектирования структуры БД завершается.

Вывод по второй главе

В результате проектирования информационной системы выделены возможные методы решения задач. Спроектированы функциональные модели системы «как есть» и «как будет». Выбраны инструментальные средства решения задачи и спроектирована база данных.

3 Реализация информационной системы

3.1 Алгоритмы распределения задач

Для магистерской диссертации использованы два основных метода распределения заявок относительно исполнителей: метод Монте-Карло и метод BPM-ранжирования.

Для метода Монте-Карло разработан следующий алгоритм, представленный на рисунке 3.1.

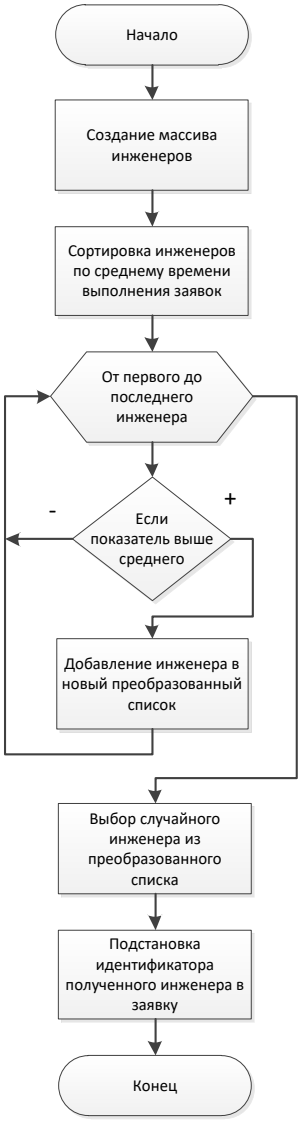


Рисунок 3.1 – Алгоритм работы метода «Монте-Карло» в информационной системе

Как видно из рисунка 3.1 при поиске инженера сначала формируется общий массив с инженерами, после чего данный список сортируется по какому-либо более общему показателю. В качестве показателя для сортировки выбрано общее время исполнения заявок каждого инженера. Отсортировав данный список, происходит фильтрация и формирование нового списка с теми инженерами, показатели которых являются выше среднего показателя из общего набора инженеров. Из полученного нового набора происходит выбор случайного инженера и подстановка его идентификатора в создаваемое обращение.

Данный метод с представленным алгоритмом удобен и полезен в тех случаях, когда организация испытывает трудности в связи с большим наплывом заявок и требуется выбрать оптимального исполнителя без наиболее детального расчета компетентных лиц.

Для более точного расчета более компетентных лиц для исполнения заявок разработан алгоритм, основанный на BPM-ранжировании. Данный алгоритм представлен на рисунке 3.2.

Как видно из рисунка 3.2 при поиске инженера сначала формируется общий массив с инженерами, после чего данный список фильтруется и выбираются только те инженеры, которые уже успешно выполняли данный тип заявок. Отфильтровав данный список, происходит расчет для каждого инженера коэффициента компетентности по формуле 3.1, идентичной представленной на блок-схеме.

$$\mathrm{K} = \frac{\mathrm{K}_{\mathrm{count}} * \mathrm{K}_{\mathrm{AverageEvaluation}}}{\mathrm{K}_{\mathrm{AverageExecutionTime}}}, \qquad (3.1)$$

где K_{count} – количество выполненных заявок данного типа;

$\mathrm{K}_{\mathrm{AverageEvaluation}}$ – средняя оценка выполнения данного типа заявок;

$\mathrm{K}_{\mathrm{AverageExecutionTime}}$ – среднее время исполнения заявок данного типа.

Исходя из полученных коэффициентов по каждому компетентному инженеру, происходит выбор инженера с наибольшим показателем коэффициента K.

Таким образом, основное отличие данного метода от рассматриваемого ранее метода Монте-Карло – это исключение выбора случайного исполнителя, путем более точного анализа и расчета коэффициента полезности относительно предыдущих выполненных работ.

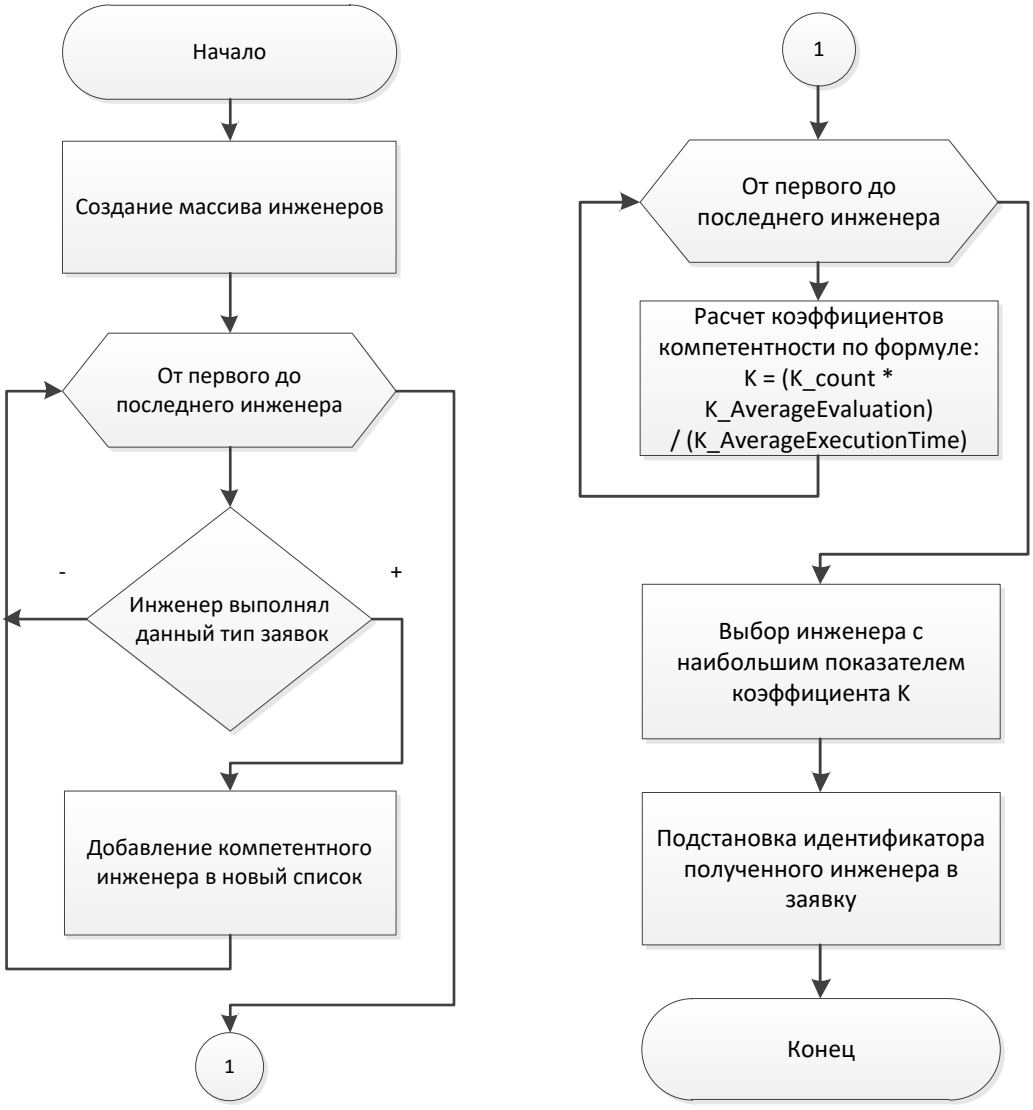


Рисунок 3.2 – Алгоритм работы разработанного метода, основанного на BPM-ранжировании

Каждый из разработанных и реализованных методов обладает особенностями и преимуществами. Распределение методом Монте-Карло

удобно и полезно в тех случаях, когда организация испытывает трудности в связи с большим наплывом заявок и требуется выбрать оптимального исполнителя без детального расчета компетентных лиц. В случае, когда требуется подобрать исполнителя более точно на основании предыдущих показателей успешных работ – имеется возможность применить распределение методом BPM-ранжирования.

При создании заявок пользователь имеет возможность выбрать одни из реализованных методов распределения заявки между исполнителями, либо исключить использование методов распределения и выбрать требуемого исполнителя вручную.

3.2 Реализация спроектированной БД

Для реализации спроектированной ранее базы данных в среде PhpMyAdmin созданы поля и таблицы, идентичные физической модели, представленной на этапе проектирования. Список созданных таблиц в среде PhpMyAdmin представлен на рисунке 3.3.

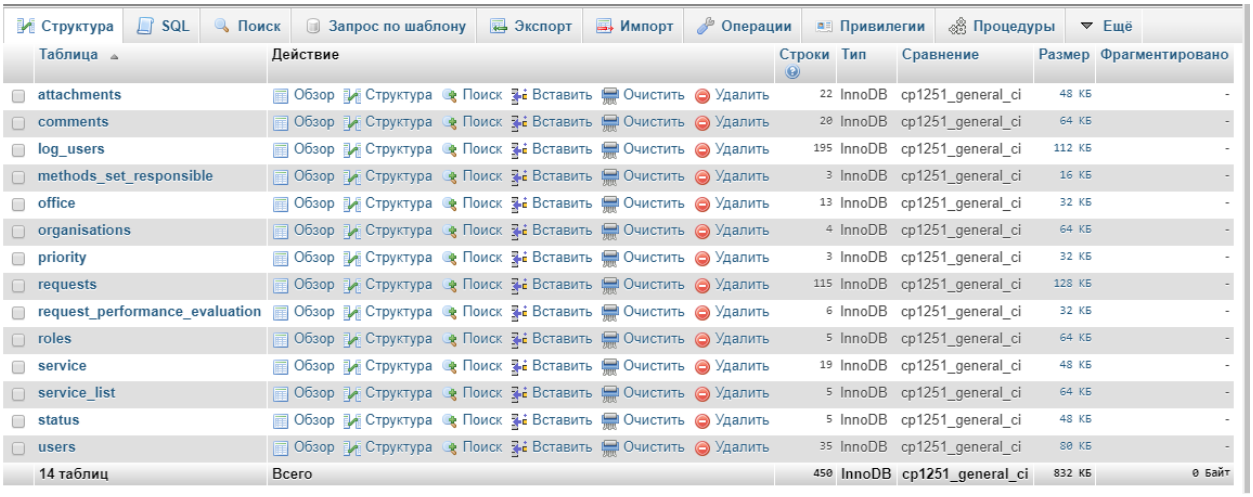


Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
attachments	Обзор Структура Поиск Вставить Очистить Удалить	22	InnoDB	cp1251_general_ci	48 КБ	-
comments	Обзор Структура Поиск Вставить Очистить Удалить	20	InnoDB	cp1251_general_ci	64 КБ	-
log_users	Обзор Структура Поиск Вставить Очистить Удалить	195	InnoDB	cp1251_general_ci	112 КБ	-
methods_set_responsible	Обзор Структура Поиск Вставить Очистить Удалить	3	InnoDB	cp1251_general_ci	16 КБ	-
office	Обзор Структура Поиск Вставить Очистить Удалить	13	InnoDB	cp1251_general_ci	32 КБ	-
organisations	Обзор Структура Поиск Вставить Очистить Удалить	4	InnoDB	cp1251_general_ci	64 КБ	-
priority	Обзор Структура Поиск Вставить Очистить Удалить	3	InnoDB	cp1251_general_ci	32 КБ	-
requests	Обзор Структура Поиск Вставить Очистить Удалить	115	InnoDB	cp1251_general_ci	128 КБ	-
request_performance_evaluation	Обзор Структура Поиск Вставить Очистить Удалить	6	InnoDB	cp1251_general_ci	32 КБ	-
roles	Обзор Структура Поиск Вставить Очистить Удалить	5	InnoDB	cp1251_general_ci	64 КБ	-
service	Обзор Структура Поиск Вставить Очистить Удалить	19	InnoDB	cp1251_general_ci	48 КБ	-
service_list	Обзор Структура Поиск Вставить Очистить Удалить	5	InnoDB	cp1251_general_ci	64 КБ	-
status	Обзор Структура Поиск Вставить Очистить Удалить	5	InnoDB	cp1251_general_ci	48 КБ	-
users	Обзор Структура Поиск Вставить Очистить Удалить	35	InnoDB	cp1251_general_ci	80 КБ	-
14 таблиц	Всего	450	InnoDB	cp1251_general_ci	832 КБ	0 Байт

Рисунок 3.3 – Список созданных таблиц в среде PhpMyAdmin

Так же для таблицы логов работы с пользователями разработаны 3 триггера. Данные триггеры представлены на рисунке 3.4.

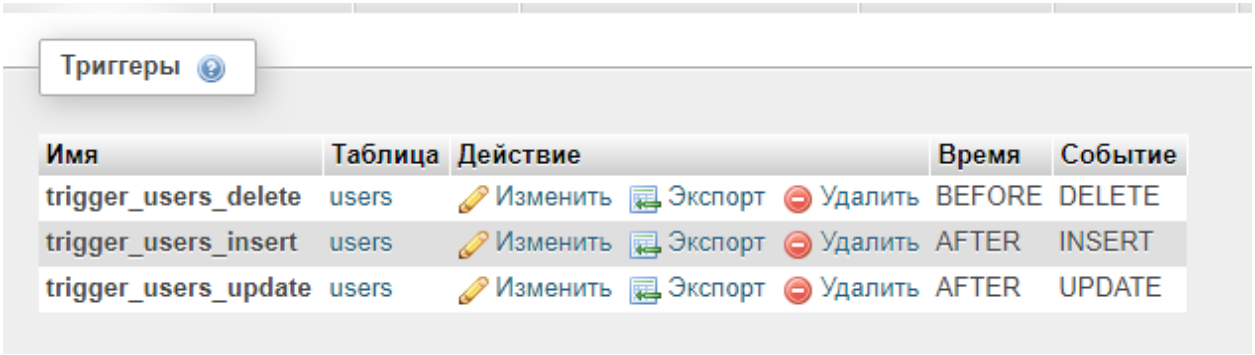
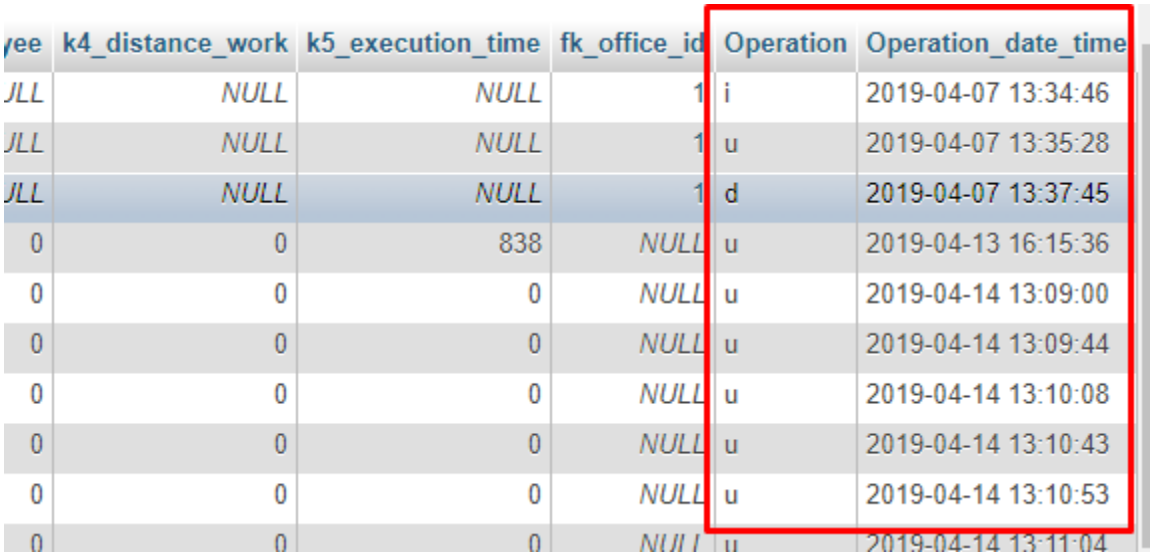


Рисунок 3.4 – Список созданных триггеров

Как видно из названий триггеров, срабатывание каждого из них завязано на одной из трех операций: удаление, создание или обновление с заданными условиями до начала или после завершения операции. Структура таблицы идентична таблице пользователей с двумя дополнительными полями: название операции и дата совершения операции. При совершении одной из операций, в данную таблицу записываются строка с идентификатором операции и дата совершения операции. Данные отличия представлены на рисунке 3.5.



...yee	k4_distance_work	k5_execution_time	fk_office_id	Operation	Operation_date_time
...JLL	NULL	NULL	1	i	2019-04-07 13:34:46
...JLL	NULL	NULL	1	u	2019-04-07 13:35:28
...JLL	NULL	NULL	1	d	2019-04-07 13:37:45
0	0	838	NULL	u	2019-04-13 16:15:36
0	0	0	NULL	u	2019-04-14 13:09:00
0	0	0	NULL	u	2019-04-14 13:09:44
0	0	0	NULL	u	2019-04-14 13:10:08
0	0	0	NULL	u	2019-04-14 13:10:43
0	0	0	NULL	u	2019-04-14 13:10:53
0	0	0	NULL	u	2019-04-14 13:11:04

Рисунок 3.5 – Поля с идентификатором операции и датой в таблице логов пользователей.

Как видно из рисунка 3.5 в качестве идентификатора операции используются 3 основных символа: «i», «u» и «d».

Данные символы обозначают следующие операции:

– символ «i» обозначает операцию insert и момент, когда данная учетная запись была добавлена в таблицу пользователей;

– символ «u» отражает промежуток, когда запись в таблице пользователей редактировалась;

– символ «d» позволяет увидеть в какой момент учетная запись была удалена.

Полагаясь на приведенные данные, администраторы смогут восстанавливать удаленные записи из базы данных в случае необходимости.

Три созданных триггера представлены на рисунках 3.6, 3.7 и 3.8. Триггер, который используется при добавлении записи в таблицу пользователей представлен на рисунке 3.6.

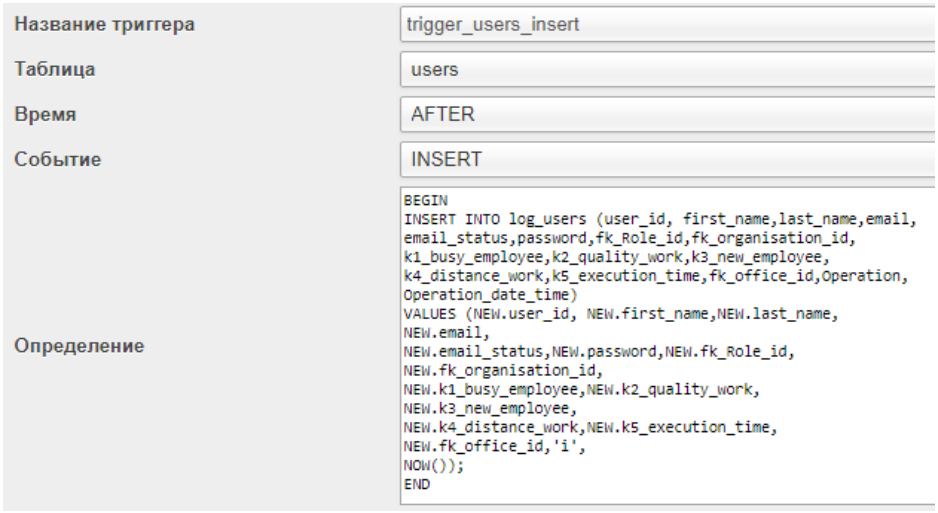


Рисунок 3.6 –Триггер на срабатывание при добавление записи в таблицу пользователей

Триггер, представленный на рисунке 3.6 срабатывает до операции добавления записи в таблицу логов.

Триггер, срабатывающий при обновлении записи в таблице пользователей представлен на рисунке 3.7. Данный триггер срабатывает после выполнения системой операции обновления записи в базе данных

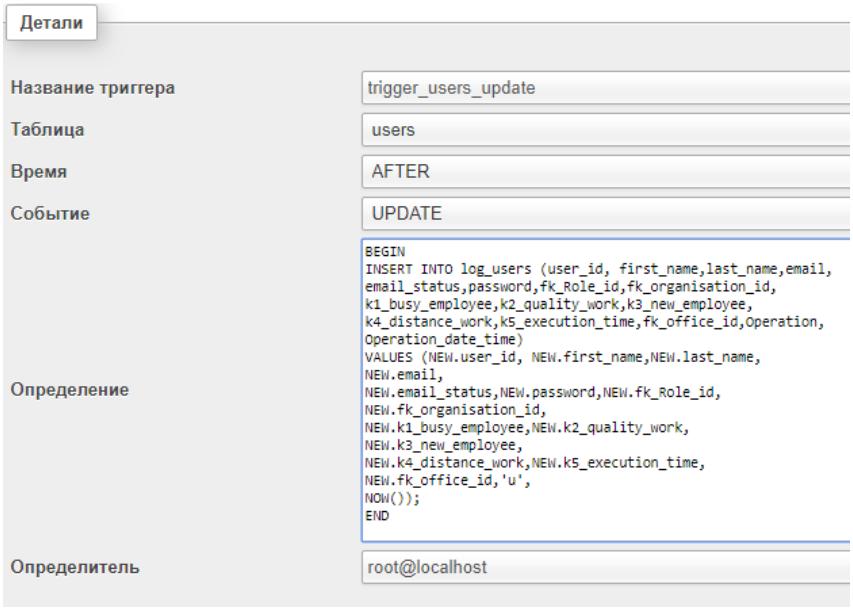


Рисунок 3.7 –Триггер на срабатывание при изменении записи в таблице пользователей

Как видно из рисунка 3.7 основной скрипт почти не отличается от скрипта на добавление записи. Основными изменениями являются параметры на срабатывание триггера и несколько других параметров.

На рисунке 3.8 представлен триггер, который срабатывает при удалении учетной записи пользователя.

Рисунок 3.8 –Триггер на срабатывание при удалении учетной записи пользователя

На рисунке 3.9 представлен дамп основной таблицы обращений (requests).

```
CREATE TABLE IF NOT EXISTS `requests` (
  `request_id` int(11) NOT NULL AUTO_INCREMENT,
  `caption` varchar(200) NOT NULL,
  `short_description` varchar(500) NOT NULL,
  `description` longtext NOT NULL,
  `fk_create_user_id` int(11) NOT NULL,
  `fk_responsible_user_id` int(11) NOT NULL,
  `date_create` datetime NOT NULL,
  `date_start_work` datetime DEFAULT NULL,
  `date_resolve` datetime DEFAULT NULL,
  `fk_status_id` int(11) NOT NULL,
  `fk_priority_id` int(11) NOT NULL,
  `fk_service_id` int(11) NOT NULL,
  `evaluation_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`request_id`),
  KEY `request_id` (`request_id`),
  KEY `fk_create_user_id` (`fk_create_user_id`),
  KEY `fk_responsible_user_id` (`fk_responsible_user_id`),
  KEY `fk_status_id` (`fk_status_id`),
  KEY `fk_priority_id` (`fk_priority_id`),
  KEY `fk_service_id` (`fk_service_id`),
  KEY `evaluation_id` (`evaluation_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=cp1251 AUTO_INCREMENT=117 ;
```

Рисунок 3.9 – Дамп таблицы обращений (requests)

Как видно из рисунка 3.9 все поля соответствуют построенной ранее физической модели базы данных. Данная таблица обладает автоинкрементируемым первичным ключом.

На рисунке 3.10 представлен дамп таблицы пользователей (users).

```
CREATE TABLE IF NOT EXISTS `users` (
  `user_id` int(11) NOT NULL AUTO_INCREMENT,
  `first_name` varchar(255) NOT NULL,
  `last_name` varchar(255) NOT NULL,
  `email` varchar(100) NOT NULL,
  `email_status` tinyint(1) NOT NULL,
  `password` varchar(100) NOT NULL,
  `fk_Role_id` int(11) NOT NULL,
  `fk_organisation_id` int(11) NOT NULL,
  `k1_busy_employee` tinyint(1) DEFAULT NULL,
  `k2_quality_work` float DEFAULT NULL,
  `k3_new_employee` tinyint(1) DEFAULT NULL,
  `k4_distance_work` float DEFAULT NULL,
  `k5_execution_time` time DEFAULT NULL,
  `fk_office_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`user_id`),
  UNIQUE KEY `email` (`email`),
  KEY `fk_Role_id` (`fk_Role_id`),
  KEY `fk_organisation_id` (`fk_organisation_id`),
  KEY `fk_office_id` (`fk_office_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=cp1251 AUTO_INCREMENT=38 ;
```

Рисунок 3.10 – Дамп таблицы обращений (requests)

Как видно из рисунка 3.10 все поля соответствуют построенной ранее физической модели базы данных. Данная таблица обладает автоинкрементируемым первичным ключом и уникальным ключом email для исключения повторных регистраций одинаковых почтовых адресов.

Дамп таблицы офисов (office) показан на рисунке 3.11.

```
CREATE TABLE IF NOT EXISTS `office` (
  `office_id` int(11) NOT NULL AUTO_INCREMENT,
  `office_name` varchar(100) NOT NULL,
  `fk_organisation_id` int(11) DEFAULT NULL,
  `is_main` tinyint(1) DEFAULT NULL,
  `mo_distance` float DEFAULT NULL,
  PRIMARY KEY (`office_id`),
  KEY `fk_organisation_id` (`fk_organisation_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=cp1251 AUTO_INCREMENT=14 ;
```

Рисунок 3.11 – Дамп таблицы офисов (office)

Как видно из рисунка 3.11 все поля соответствуют построенной ранее физической модели базы данных. Данная таблица обладает автоинкрементируемым первичным ключом.

На рисунке 3.12 представлен дамп таблицы организаций (organisations).

```
CREATE TABLE IF NOT EXISTS `organisations` (
  `id_organisation` int(11) NOT NULL AUTO_INCREMENT,
  `organisation_name` varchar(100) NOT NULL,
  `organisation_adres` varchar(100) NOT NULL,
  `fk_main_office_id` int(11) DEFAULT NULL,
  `CountOffices` int(11) DEFAULT NULL,
  PRIMARY KEY (`id_organisation`),
  KEY `id_organisation` (`id_organisation`),
  KEY `fk_main_office_id` (`fk_main_office_id`),
  KEY `fk_main_office_id_2` (`fk_main_office_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=cp1251 AUTO_INCREMENT=4 ;
```

Рисунок 3.12 – Дамп таблицы организаций (organisations)

Все представленные рисунки с дампами таблиц полностью идентичны представленной ранее физической модели разработанной базы данных. Каждая таблица обладает первичным ключом и рядом вторичных ключей со связями один ко многому.

3.3 Разработка и тестирование информационной системы

Первым этапом при начале работы с системой требуется авторизация пользователя. Авторизация в разрабатываемой системе требуется для определение роли авторизующегося пользователя (пользователь, инженер, менеджер или администратор). В зависимости от обладаемой роли пользователю предоставляется только доступный для его роли функционал. Все данные пользователя хранятся в базе данных в зашифрованном виде в MD5 для защиты от взлома. При этом для безопасности для каждого пароля добавляется специальная ключевая строка – так называемая «соль». Соль (или модификатор) – это строка данных, которая передаётся хеш-функции вместе с паролем. Главным образом данная строка используется для защиты от перебора по словарю и атак с использованием радужных таблиц, а также для сокрытия возможных одинаковых паролей. Однако соль не может полностью защитить от полного перебора каждого отдельного пароля. Но позволяет усложнить данный способ.

Одна из важнейших задач соли – это сделать разными «хеши» паролей в случае, если два пользователя указали одинаковый пароль, тем самым усложнив перебор. Так же это актуально при условии, если одному человеку позволено иметь несколько профилей.

Реализованная форма авторизации пользователя представлена на рисунке 3.13.

Как видно из рисунка 3.13, основным критерием для входа пользователя является правильный ввод электронной почты, пароля и проверочного кода. Кроме этого учетная запись пользователя не должна иметь признак «отключенной» в базе данных. По умолчанию при создании учетных записей проставляется признак включения. Однако администраторы по каким-либо причинам могу произвести отключение.

В случае если одно из критериев не выполнено – будет выдано соответствующее текстовое сообщение с ошибкой.

В случае попытки перехода на страницы основной работы с системой без авторизации – выдаются информационные сообщения о том, что пользователь зашел на страницу напрямую и требуется авторизация.

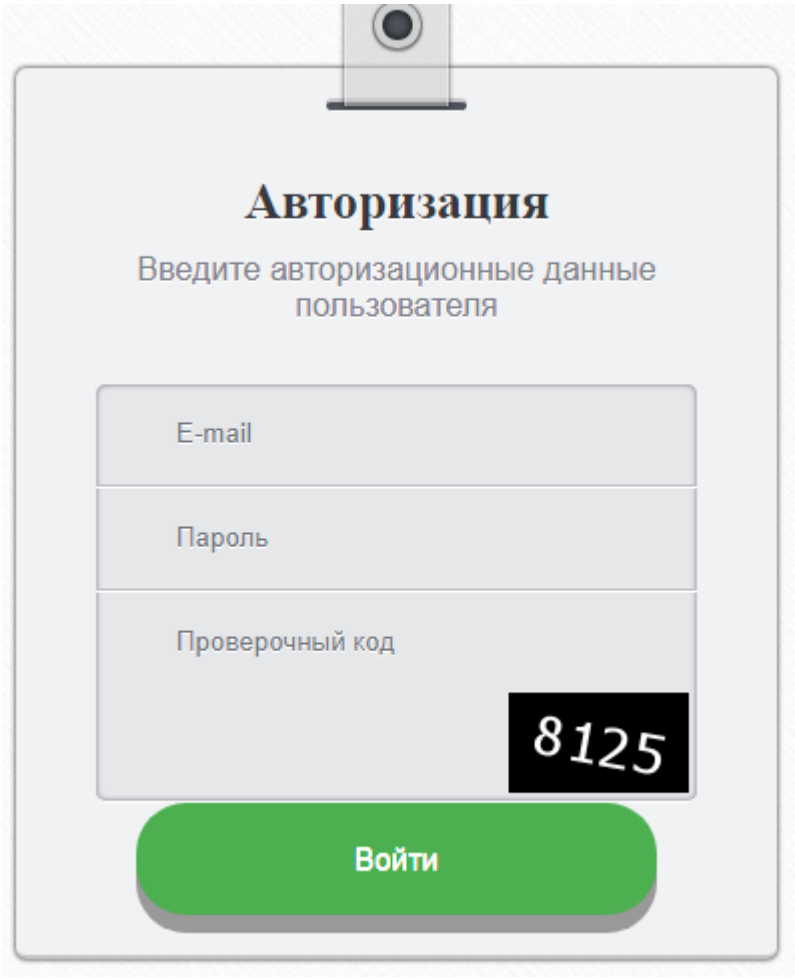


Рисунок 3.13 – Форма авторизации пользователя

После успешной авторизации, на основной навигационной панели отображаются основные возможные действия, представленные на рисунке 3.14.

Рисунок 3.14 – Форма авторизации пользователя

Так как вход был произведен под пользователем с ролью «администратор», то доступны все возможные функции.

При нажатии на кнопку «главная» пользователь попадает на начальную страницу системы. Нажав на кнопку «обращения» или «создать обращение» пользователь перенаправляется на список всех обращений в системе, либо на форму заведения нового обращения в систему соответственно.

Кнопка регистрации новых пользователей доступна только для роли «администратор». Вид данной формы представлен на рисунке 3.15.

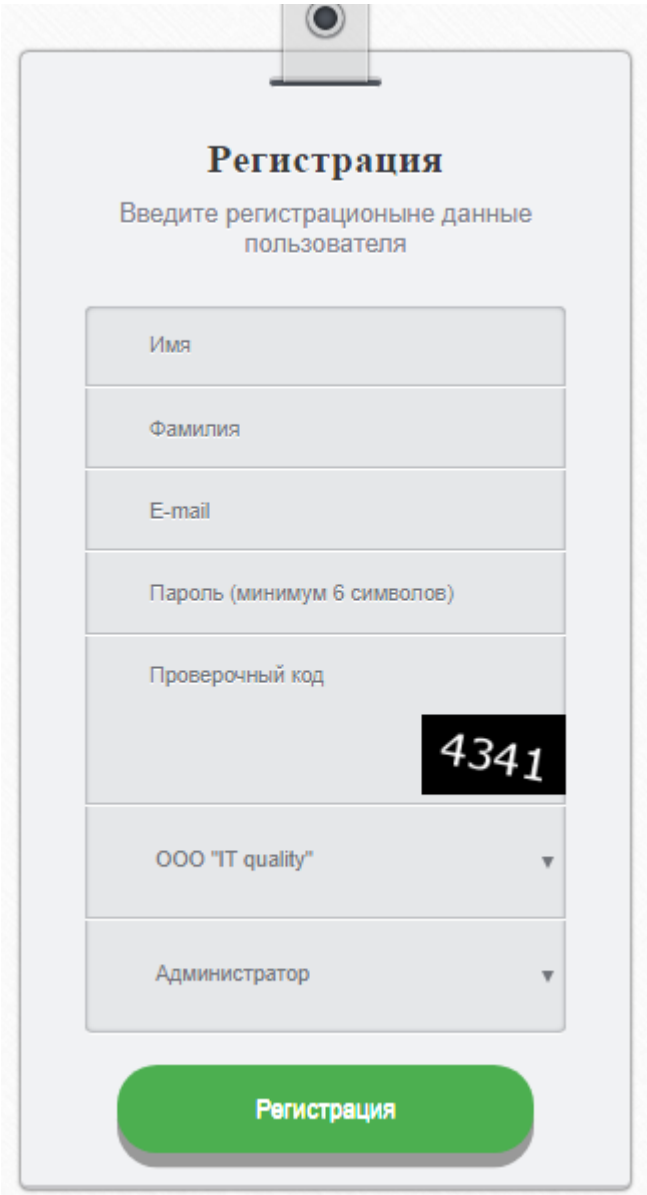


Рисунок 3.15 – Форма авторизации пользователя

Как видно из рисунка 3.15 для регистрации пользователя требуется ввести следующие данные: имя, фамилия, электронная почта, пароль, проверочный код, организация и роль пользователя. Кроме это для всех полей реализована «валидация» как на стороне клиента, так и на стороне сервера. «Валидация» производится не только в момент отправки, но и во время ввода данных, через AJAX. Это представлено на рисунке 3.16.

Аналогичным образом реализована «валидация» на форме авторизации пользователя. В случае если по каким-то причинам на стороне клиента все данные были «провалидированы» успешно, а сервер при проверке определил некорректные данные – система выдаст сообщение и отменит операцию.

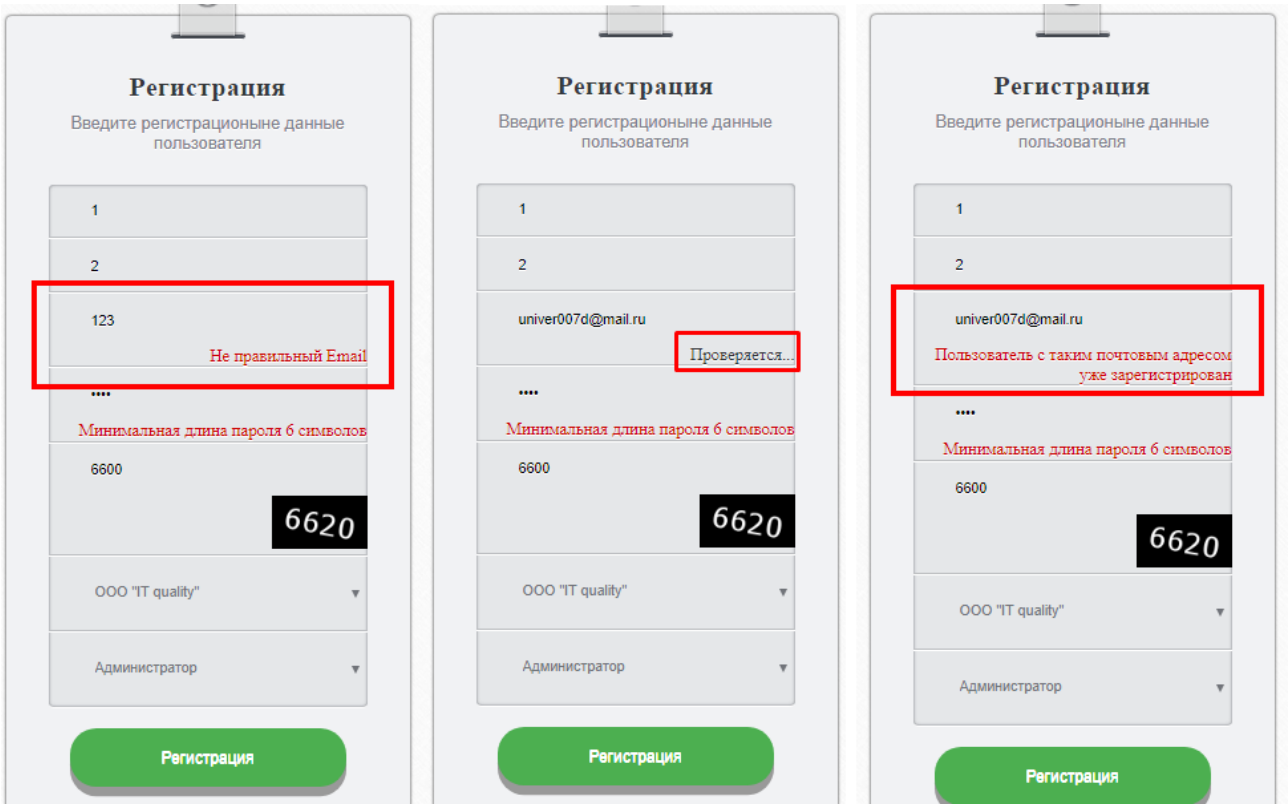


Рисунок 3.16 – Реализованная валидация для формы регистрации

На рисунке 3.17 представлена основная страница по работе с обращениями. Листинг кода, использованный для данной страницы, представлен в приложении А.

Рисунок 3.17 – Основная страница по работе с обращениями

Как видно из рисунка 3.17 форма обращений имеет фильтр, нажав на который пользователю разворачивается форма с параметрами фильтрации. Ниже формы фильтрации представлена таблица со списком обращений с возможностью сортировки каждого столбца путем клика на одну из стрелок у каждого заголовка.

Форма фильтрации обращений представлена на рисунке 3.18.

Как видно из рисунка 3.18 форма фильтрации сгруппирована на 4 основных области:

– область фильтрации по заголовкам с фильтрами по номеру, заголовку или краткому описанию заявки;

– область с фильтром по пользователям: создателю и исполнителю;

– область с фильтрами по датам: дата создания и дата решения;

– область фильтра по дополнительным данным: статуса, приоритету и типу проблемы или услуги с возможным выбором нескольких пунктов из списка.

Все выбранные параметры фильтрации хранятся в глобальном массиве с учетом сессии и действуют до сброса фильтра пользователем, изменении параметров фильтрации или до окончания сессии в браузере.

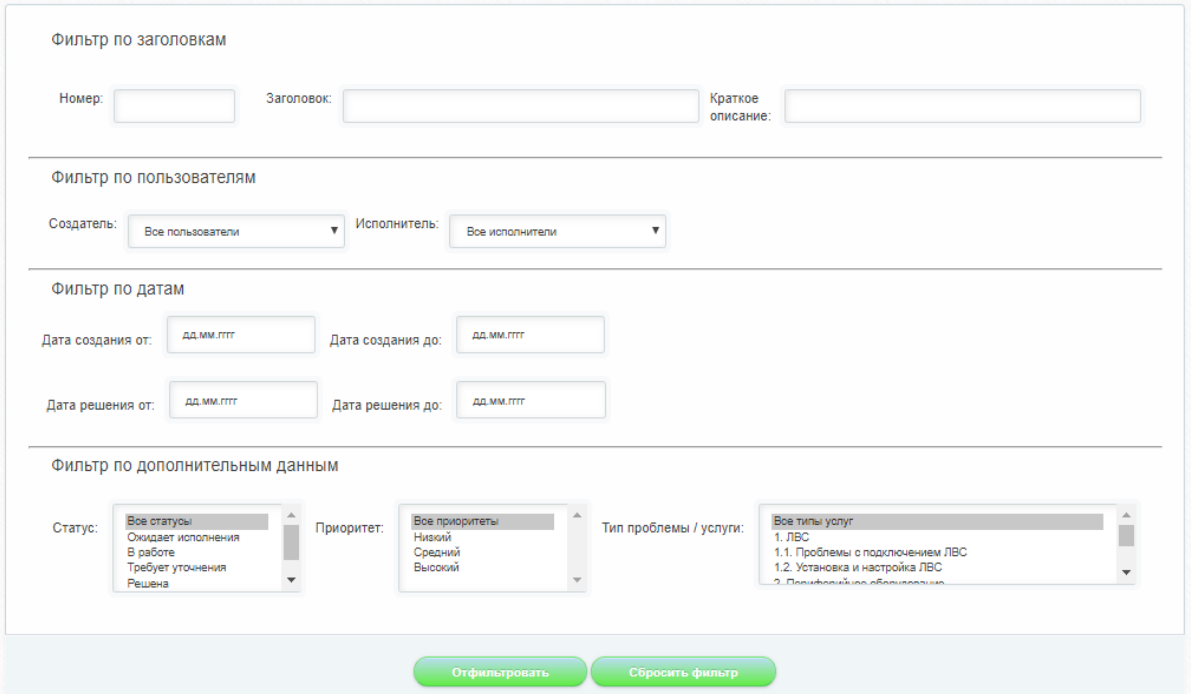


Рисунок 3.18 – Форма фильтрации обращений

Форма создания обращений представлена на рисунке 3.19. Для создания обращения пользователю требуется указать основные обязательные критерии:

– заголовок;

– краткое описание проблемы;

– полное описание проблемы;

– желаемый исполнитель;

– приоритет;

– тип проблемы или услуги;

– метод распределения исполнителя.

Поле заголовка отображает основной и исчерпывающий смысл заявки в нескольких словах.

Краткое описание требуется для отражения проблемы заявки исходя из заголовка, раскрывая более подробно суть проблемы в одном или нескольких предложениях.

Полное описание проблемы подразумевает полное и обширное раскрытие сути проблемы с указанием всех возможных подробностей.

Поле желаемый исполнитель используется в случае если пользователь выбрал условие выбора инженера без автоматического распределения по предоставленным системой алгоритмам в последнем пункте на представленном ранее списке.

Приоритет отражает приоритет важности заявки: низкий, средний или высокий.

Тип проблемы или услуги позволяет пользователю выбрать из выпадающего списка тот тип проблемы, к которому относится создаваемая задача.

Метод распределения исполнителя отвечает за алгоритм, который будет использован при назначении инженера на заявку. На данный момент пользователь имеет возможность не использовать предоставленные методы и назначить исполнителя самому, либо использовать методы «Монте-Карло» или «BPM-ранжирования» для оптимального назначения исполнителя создаваемой заявке.

Помимо заполняемых данных, пользователь имеет доступ к прикреплению нескольких файлов в обращении. Данная возможность позволяет прикреплять скриншоты ошибок или другие файлы, помогающие инженерам более детально изучить проблему и быстрее приступить к работе. В целях безопасности, ограничен так же и набор типов файлов, которые может прикрепить пользователь для избегания возможных попыток взлома через скрипты и прочие способы. Процесс прикрепления файлов приведен на рисунке 3.20.

При прикреплении файлов все имена переводится на латиницу в целях исключения ошибок при работе с кириллическими символами.

Так же при загрузке файлов с одинаковыми наименованиями производится приписывание к имени файла номера копии для исключения перезаписи уже имеющегося файла.

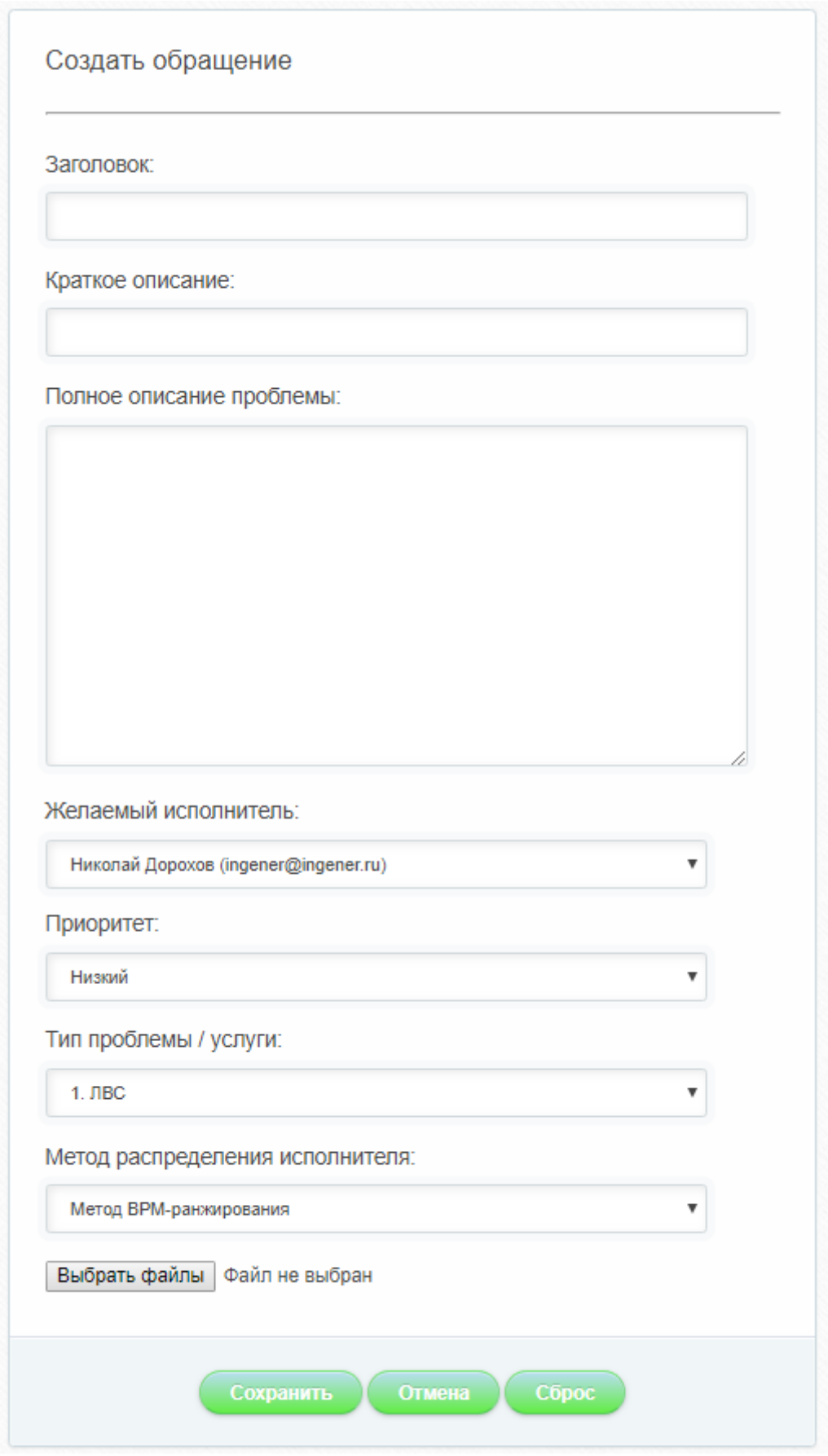


Рисунок 3.19 – Форма создания обращений

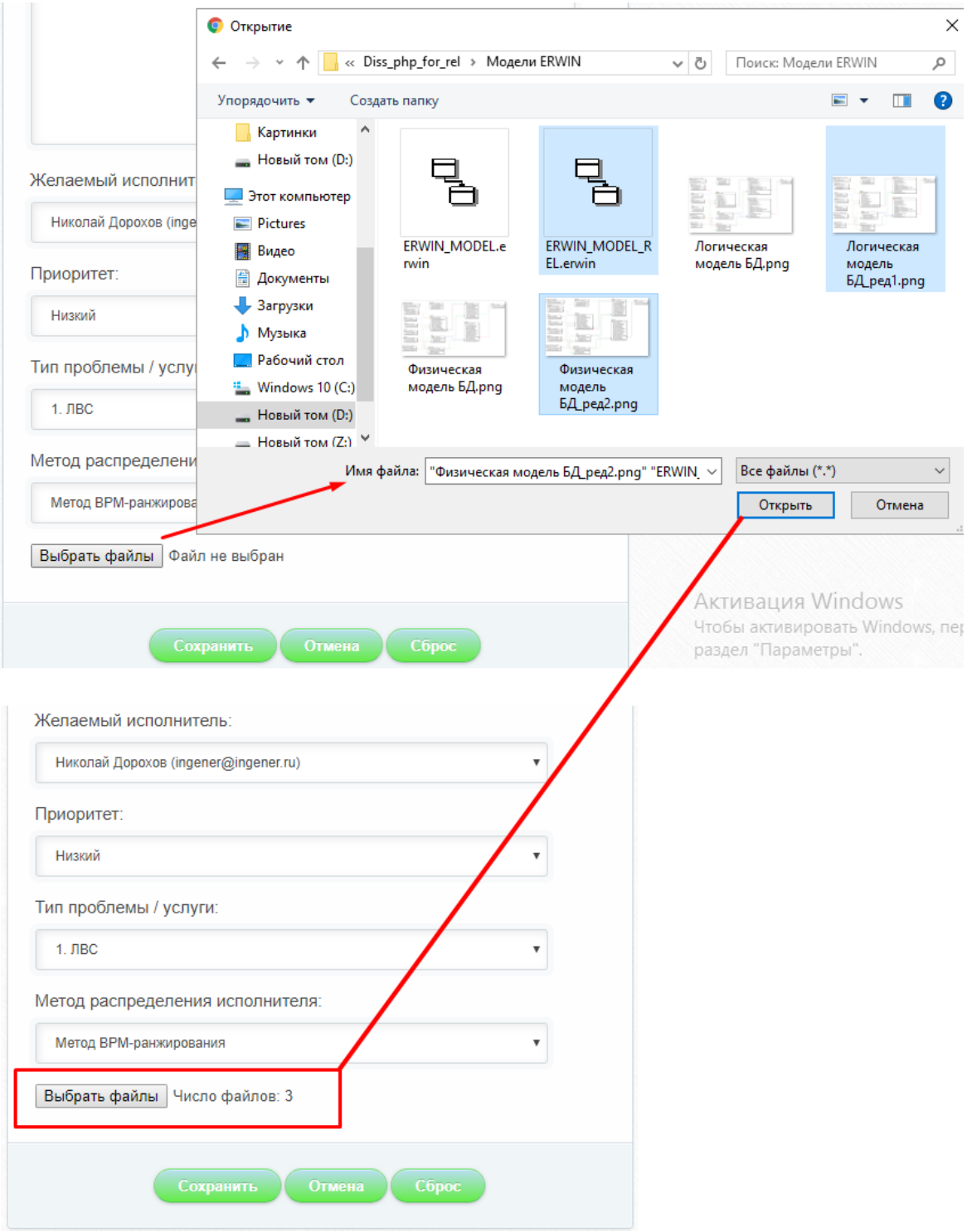


Рисунок 3.20 – Прикрепление файлов в обращении

Создав обращение кнопкой сохранить, пользователь имеет возможность перейти к просмотру и редактированию обращения. Окончательный вид после заполнения всех полей формы создания обращения представлен на рисунке 3.21.

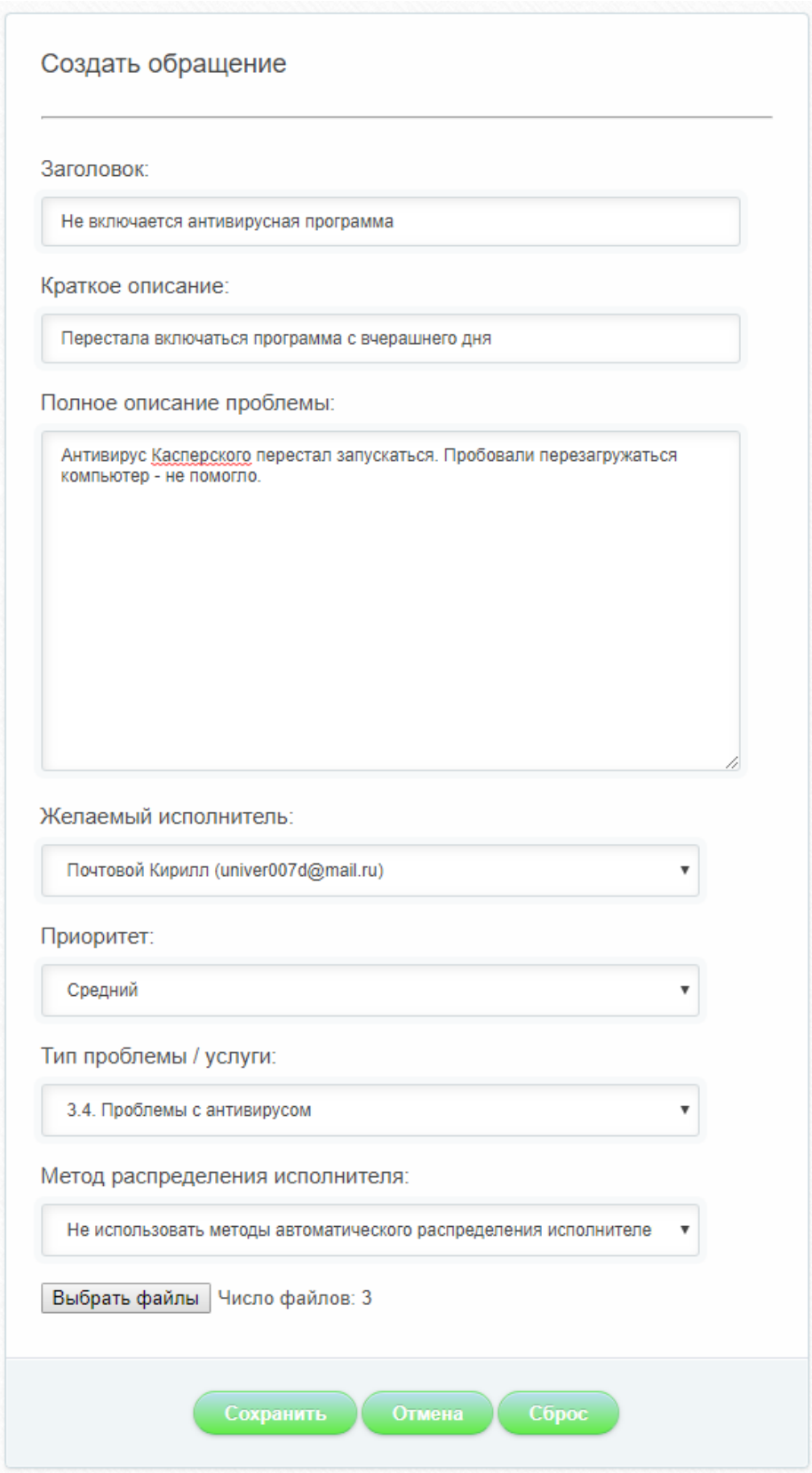


Рисунок 3.21 – Окончательный вид после заполнения всех полей формы создания обращения

Как видно из рисунка 3.21 пользователь отказался от использования методов автоматического распределения исполнителей и сам назначил в качестве исполнителя пользователя «Почтовой Кирилл» с электронной почтой «univer007d@mail.ru». После сохранения на несколько секунд появляется сообщение об успешном создании обращения и созданная заявка отображается в списке для текущего пользователя.

Список заявок для текущего пользователя представлен на рисунке 3.22.

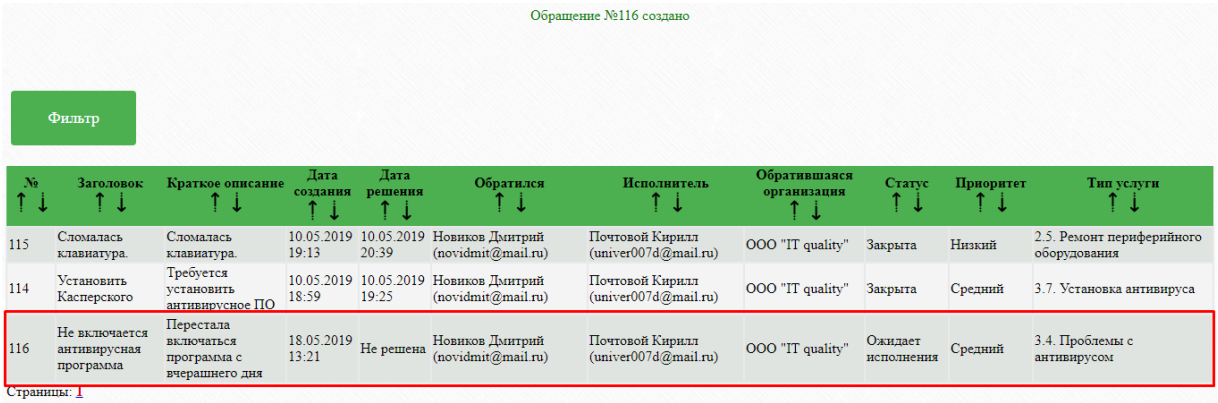


Рисунок 3.22 – Список заявок для текущего пользователя

Вне зависимости от использованных методов распределения заявок относительно инженеров каждому назначенному исполнителю – инженеру приходит письмо от системы с уведомлением на электронную почту. Вид данного письма для созданной заявки по рисунку 3.21 представлен на рисунке 3.23.

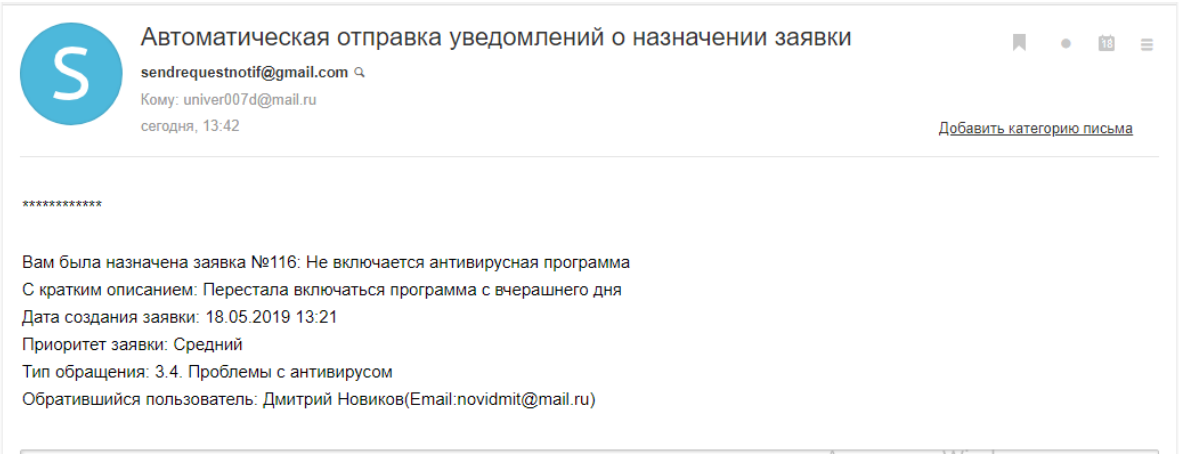


Рисунок 3.23 – Вид письма для инженера при назначении заявки

При авторизации инженера-исполнителя «Почтовой Кирилл» (по логину «univer007d@mail.ru») в системе, отобразится так же список только тех задач, в которых он указан в качестве исполнителя. Данный список с созданной ранее заявкой представлен на рисунке 3.24.

№	Заголовок	Краткое описание	Дата создания	Дата решения	Обратился	Исполнитель	Обратившаяся организация	Статус	Приоритет	Тип услуги
116	Не включается антивирусная программа	Перестала включаться программа с вчерашнего дня	18.05.2019 13:21	Не решена	Новиков Дмитрий (novidmit@mail.ru)	Почтовой Кирилл (univer007d@mail.ru)	ООО "IT quality"	Ожидает исполнения	Средний	3.4. Проблемы с антивирусом
115	Сломалась клавиатура.	Сломалась клавиатура.	10.05.2019 19:13	10.05.2019 20:39	Новиков Дмитрий (novidmit@mail.ru)	Почтовой Кирилл (univer007d@mail.ru)	ООО "IT quality"	Закрыта	Низкий	2.5. Ремонт периферийного оборудования
114	Установить Касперского	Требуется установить антивирусное ПО	10.05.2019 18:59	10.05.2019 19:25	Новиков Дмитрий (novidmit@mail.ru)	Почтовой Кирилл (univer007d@mail.ru)	ООО "IT quality"	Закрыта	Средний	3.7. Установка антивируса
113	Подключить сетевую периферию	Подключить сетевую периферию	03.05.2019 15:36	0000-00-00 00:00:00	Новиков Дмитрий (admin@admin.ru)	Почтовой Кирилл (univer007d@mail.ru)	ООО "Информ инк"	Требует уточнения	Высокий	2.2. Подключение к ЛВС сетевой периферии
112	Подключить сетевую периферию	Подключить сетевую периферию	03.05.2019 15:31	Не решена	Новиков Дмитрий (admin@admin.ru)	Почтовой Кирилл (univer007d@mail.ru)	ООО "Информ инк"	Ожидает исполнения	Высокий	2.2. Подключение к ЛВС сетевой периферии

Рисунок 3.24 – Список заявок в исполнении для инженера Почтовой Кирилл (univer007d@mail.ru)

При нажатии на одну из заявок инженер имеет возможность подробнее ознакомиться с обращением, просмотреть и скачать содержимое прикрепленных файлов, а так же в случае необходимости оставить свой комментарий к задаче. Кроме того, инженер не имеет доступа к редактированию основных исходных полей формы назначенной заявки: заголовку, краткому описанию и полному описанию. Однако обладает возможностью менять другие важные для инженеров параметры.

Вид открытого на редактирование обращения представлен на рисунке 3.25.

При сохранении изменений каких-либо параметров заявки или написания комментария пользователю создавшего заявку и исполнителю приходит письмо от системы с уведомлением, что заявка была изменена или был добавлен комментарий.

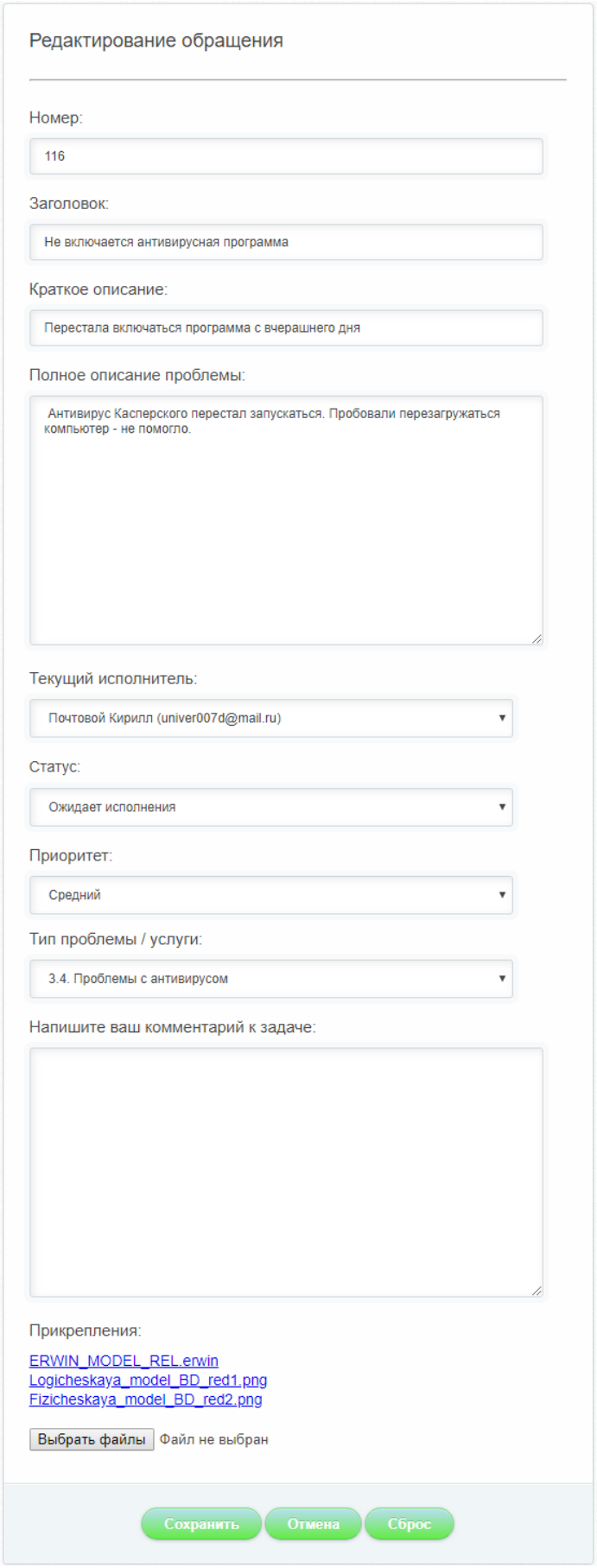


Рисунок 3.25 – Вид открытого на редактирование обращения

Вид письма, отправляемого при создании и редактировании заявки, представлен на рисунке 3.26.

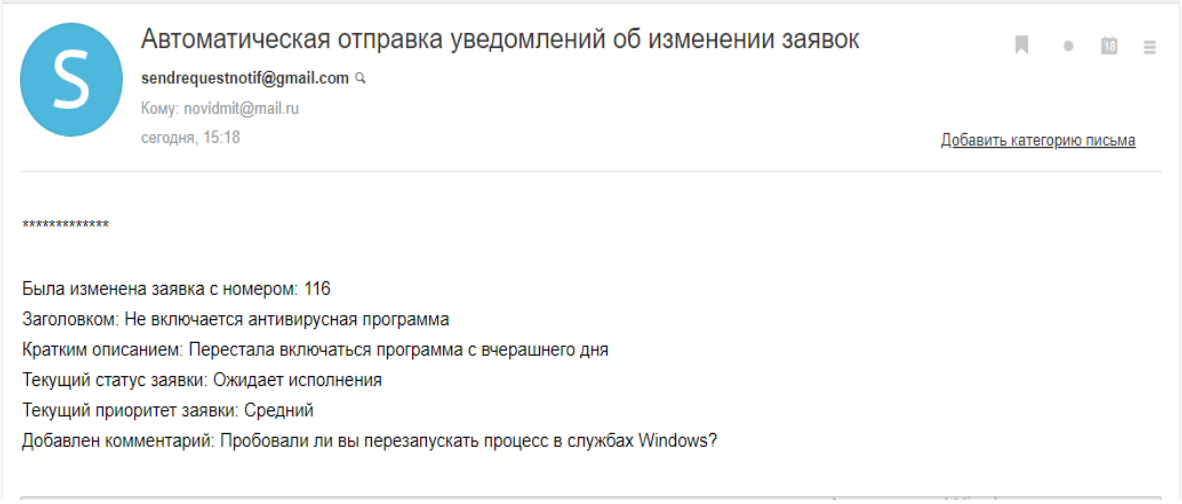


Рисунок 3.26 – Вид письма с уведомлением об изменении заявки

Письма для субъекта обращения и исполнителя имеют идентичное содержание и формат.

Добавленные комментарии к заявке отображаются ниже основной формы редактирования. Вид формы с комментариями представлен на рисунке 3.27.

Рисунок 3.27 – Форма с комментариями заявки

Каждая заявка в системе от создания до окончательного решения должна пройти 4 основных этапа, которые фиксируются промежуточными статусами. Основными статусами являются:

– ожидает исполнения;

– в работе;

– решена;

– закрыта.

Статус «ожидает исполнения» ставится у только что созданных заявок до начала работы, а статус «в работе» обозначает, что работы по данной заявке ведутся текущим исполнителем. В случае возникновения проблем или неточностей по заявке инженер может перевести заявку в статус «требует уточнения» информируя тем самым обратившегося пользователя о том, что предоставленных исходных данных в заявке не достаточно для решения задачи.

Заявка переводится в статус «решена» если все проблемы и задачи по обращению были решены и требуют проверки со стороны субъекта обращения.

Закрытая заявка в статусе «закрыта» обозначает что проблема и задачи по обращению решены и проверены обеими сторонами.

При закрытии заявки пользователь создавший обращение имеет возможность выставить оценку исполнения, которая повлияет на общий показатель исполнителя при распределении заявок. Процесс отображения таких элементов реализован через AJAX, не требуя перезагрузки страницы. В качестве критерия оценивания исполнения заявок используется пятибалльная система со следующими наименованиями и баллами:

– без указания оценки – 0 баллов;

– очень плохо – 1 балл;

– плохо – 2 балла;

– удовлетворительно – 3 балла;

– хорошо – 4 балла;

– отлично – 5 баллов.

Процесс проставления оценки исполнения заявки представлен на рисунке 3.28.

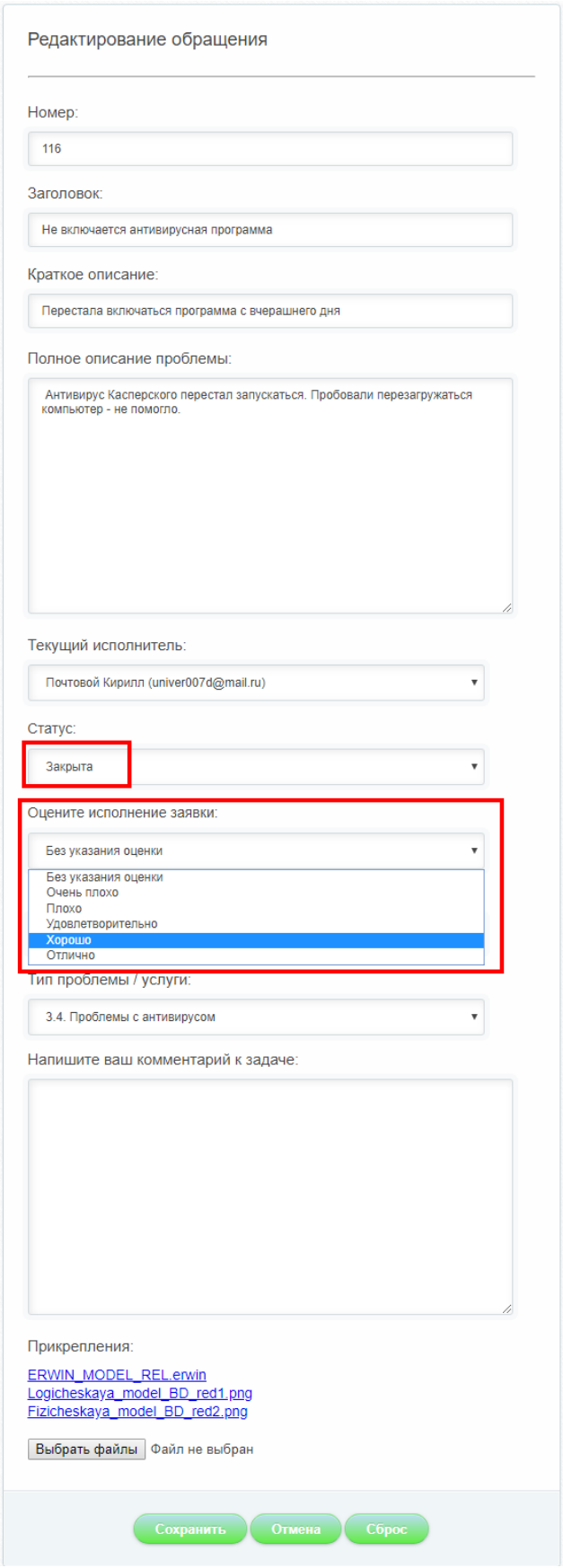


Рисунок 3.28 – Закрытие заявки с указанием оценки

После закрытия обращения, редактирование становится невозможным. Однако при возникновении каких-либо трудностей или в случае неточностей и неполным решением обращения, существует возможность «переоткрытия» обращения с переводом в статус «ожидает исполнения». Данная операция выполняется по нажатию кнопки «переоткрыть». В таком случае заявка потребует повторного решения проблем со стороны исполнителя и прохождения полного цикла работы с пошаговыми переходами в статусы.

При этом после закрытия какой-либо заявки инженерам-исполнителям пересчитываются и заносятся два важных коэффициента в базу данных:

– средняя оценка по выполненным задачам;

– среднее время, затраченное на выполнение задач.

С помощью данных показателей менеджеры имеются возможность следить за качеством исполняемых инженерами задач. Кроме этого данные показатели учитываются при распределении инженеров на задачи. Чем лучше коэффициент – тем компетентнее инженер в решении той или иной задачи.

Помимо этого, в параметры инженеров записывается коэффициент отражающий удаленность от места выполнения заявки. Данный коэффициент отображает числовое значение километража текущего места исполнения задачи от головного офиса исполнителя.

Так как в таблице офисов имеются занесенные значения офисов компании, а каждый инженер привязан к своему офису, то при выполнении заявок, происходит расчет расстояний на основании этих данных.

Таким образом, обладая данными о расстоянии мест удаленности работы, появляется возможность решить задачу коммивояжера

Вид формы с закрытой заявкой представлен на рисунке 3.29. На данном рисунке так же виден список прикрепленных файлов, кликнув на любой из которых появляется возможность их просмотра или сохранения.

При закрытом статусе обращения система не реагирует на попытки нажатия на редактируемые поля формы, делая их неактивными.

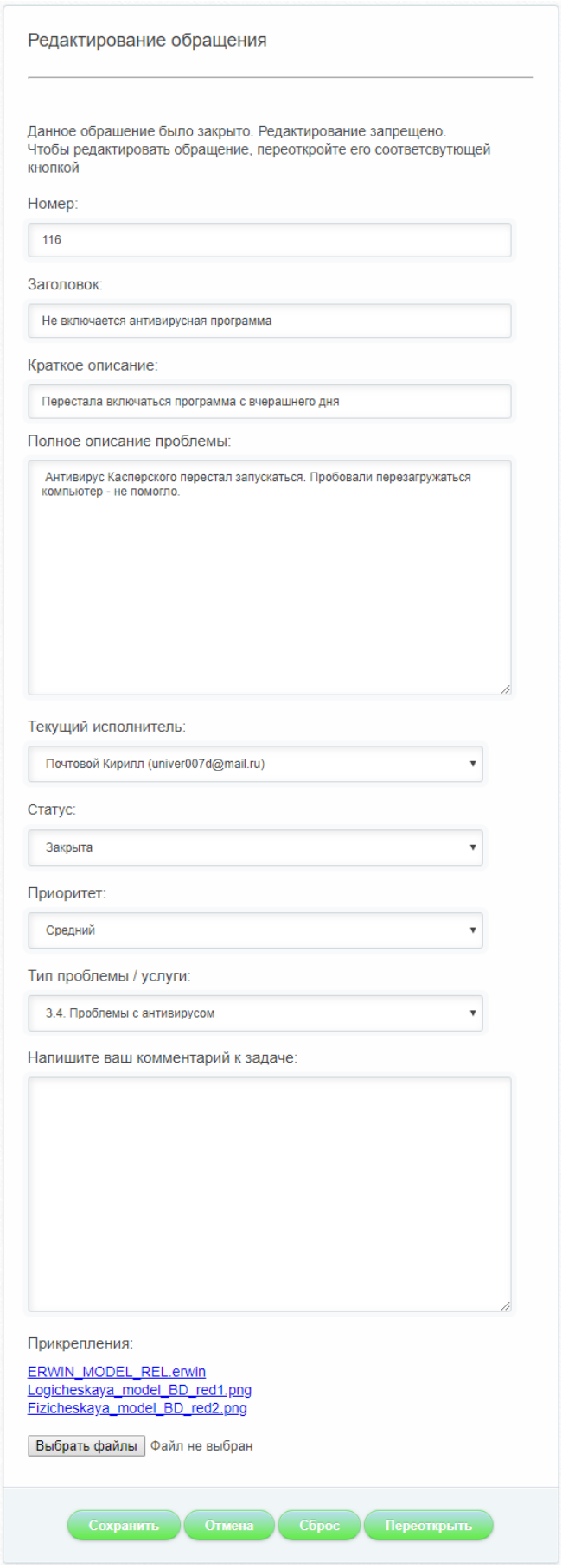


Рисунок 3.29 – Вид формы закрытой заявки

Таким образом, разработан и протестирован программный интерфейс веб-системы на языке PHP, который взаимодействует с базой данных MySQL, решая задачи по подаче и обработке обращений в IT-аутсорсинге.

3.4 Сравнительный анализ реализованных методов распределения заявок

Для проведения сравнительного анализа реализованных методов разобраны два случая при распределении заявок относительно исполнителей: распределение заявок методами при большом объеме поступающих заявок и при нормированном поступлении. Для этих целей были взяты несколько случайных инженеров, построены таблицы эффективности каждого случая распределения и их графики.

В таблице 3.1 представлены показатели распределения заявок методом «Монте-Карло» при нормированном поступлении заявок.

Таблица 3.1 – Показатели распределения заявок методом «Монте-Карло» при нормированном поступлении заявок

Показатели инженеров при распределении задач методом Монте-Карло					
ФИО	Почта	Количество заявок	Среднее время исполнения	Средняя оценка	Коэффициент
Дмитрий Новиков	Dmitry Novikov@ dn.ru	11	1:30:11	5	0,46
Дмитрий Эмальтов	emailTest @mail.ru	15	2:00:42	4	0,37
Кирилл Почтовой	univer007d @mail.ru	25	2:02:38	4,6	0,72
Анатолий Попов	ti@ti.ru	16	2:52:43	3,5	0,24
Владислав Татаринов	Test4 @Test4.ru	17	3:52:11	4	0,22
Дорохов Николай	Ingener@ ingener.ru	19	3:59:59	3,3	0,20

В таблице 3.2 представлены показатели распределения заявок методом «BPM-ранжирования» при нормированном поступлении заявок.

Таблица 3.2 – Показатели распределения заявок методом «BPM-ранжирования» при нормированном поступлении заявок

Показатели инженеров при распределении задач методом BPM-ранжирования					
ФИО	Почта	Количество заявок	Среднее время исполнения	Средняя оценка	Коэффициент
Дмитрий Новиков	Dmitry Novikov @dn.ru	11	1:15:17	4,9	0,55
Дмитрий Эмальтов	emailTest @mail.ru	15	1:45:31	4,3	0,46
Кирилл Почтовой	univer007d @mail.ru	25	1:12:33	4,9	1,29
Анатолий Попов	ti@ti.ru	16	1:59:11	3,4	0,34
Владислав Татаринов	Test4 @Test4.ru	17	2:30:16	4,3	0,36
Дорохов Николай	Ingener@ ingener.ru	19	2:33:12	3,2	0,30

Исходя из построенных таблиц нормированного поступления заявок, построен график, который отображает эффективность выполнения заявок при таком распределении. Данный график представлен на рисунке 3.30.

Исходя из построенного графика на рисунке 3.30 можно утверждать что при нормированном поступлении заявок наиболее эффективным распределением заявок относительно инженеров обладает метод BPM ранжирования. Данный метод так же довольно эффективен по сравнению с показателями до использования методов распределения исполнителей.

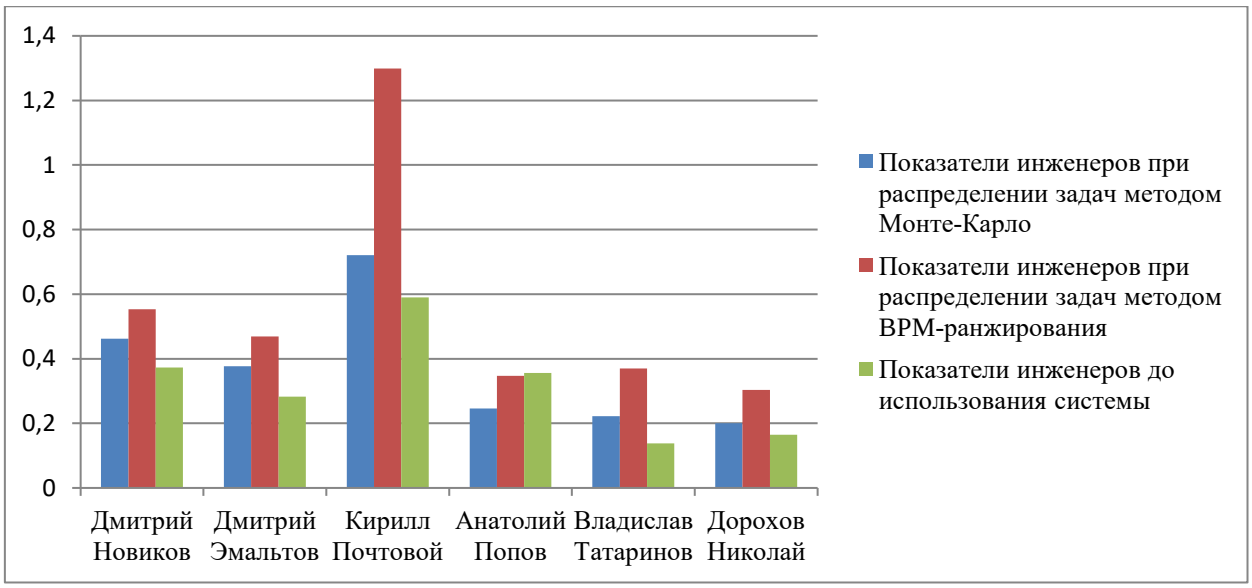


Рисунок 3.30 – График эффективности выполненных заявок с использованием разных методов распределения при нормированном поступлении заявок

Далее аналогично произведен анализ распределения методами при больших объемах поступления заявок.

В таблице 3.3 представлены показатели распределения заявок методом «BPM-ранжирования» при большом объеме поступления заявок.

Таблица 3.3 – Показатели распределения заявок методом «Монте-Карло» при большом объеме поступлении заявок

Показатели инженеров при распределении задач методом Монте-Карло					
ФИО	Почта	Количество заявок	Среднее время исполнения	Средняя оценка	Коэффициент
Дмитрий Новиков	Dmitry Novikov @dn.ru	27	1:30:11	5	1,13
Дмитрий Эмальтов	emailTest @mail.ru	37	2:00:42	4	0,92
Кирилл Почтовой	univer007d @mail.ru	62	2:02:38	4,6	1,78

Продолжение таблицы 3.3

Показатели инженеров при распределении задач методом Монте-Карло					
ФИО	Почта	Количество заявок	Среднее время исполнения	Средняя оценка	Коэффициент
Анатолий Попов	ti@ti.ru	40	2:52:43	3,5	0,61
Владислав Татаринов	Test4 @Test4.ru	42	3:52:11	4	0,54
Дорохов Николай	Ingener@ ingener.ru	47	3:59:59	3,3	0,49

В таблице 3.4 представлены показатели распределения заявок методом «BPM-ранжирования» при большом объеме поступления заявок.

Таблица 3.4 – Показатели распределения заявок методом «BPM-ранжирования» при большом объеме поступления заявок

Показатели инженеров при распределении задач методом BPM-ранжирования					
ФИО	Почта	Количество заявок	Среднее время исполнения	Средняя оценка	Коэффициент
Дмитрий Новиков	Dmitry Novikov @dn.ru	27	1:31:17	4,5	1,01
Дмитрий Эмальтов	emailTest @mail.ru	37	1:49:31	4,1	1,06
Кирилл Почтовой	univer007d @mail.ru	62	2:12:33	4,2	1,49
Анатолий Попов	ti@ti.ru	40	2:59:11	3,4	0,58
Владислав Татаринов	Test4 @Test4.ru	42	2:55:16	4,1	0,74
Дорохов Николай	Ingener@ ingener.ru	47	3:33:12	3,2	0,54

Далее аналогично построен график, который отображает эффективность выполнения заявок при таком распределении. Данный график представлен на рисунке 3.31.

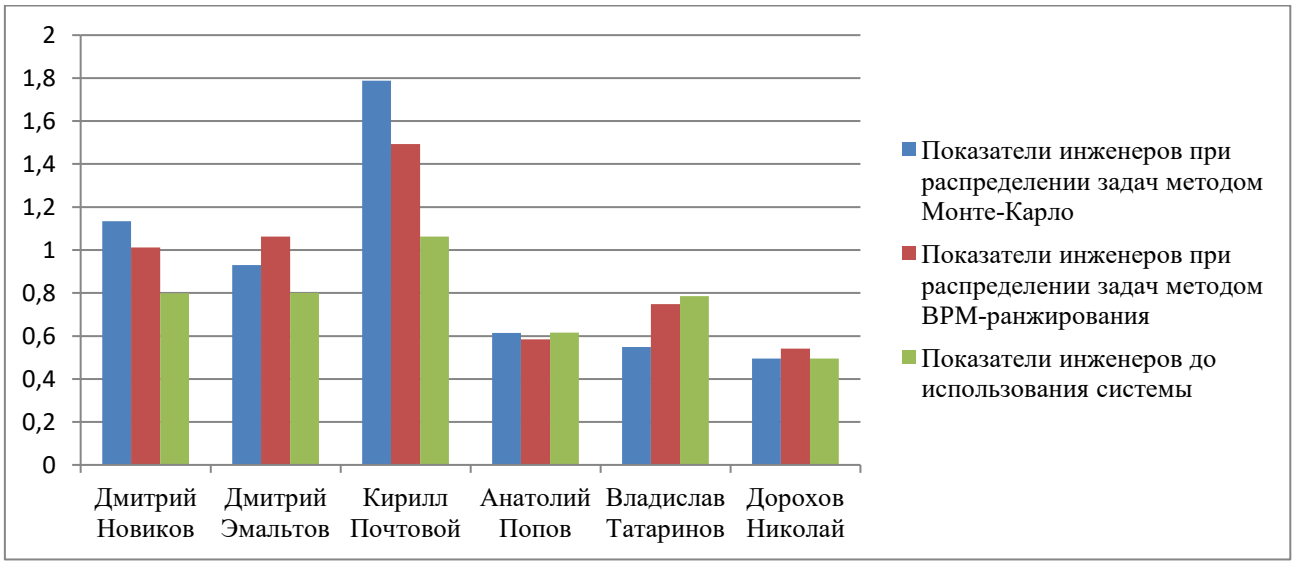


Рисунок 3.31 – График эффективности выполненных заявок с использованием разных методов распределения при большом объеме поступления заявок

Исходя из построенного графика на рисунке 3.31 можно утверждать, что при большом объеме поступления заявок эффективность использования метода «ВРМ-ранжирования» по сравнению с методом «Монте-Карло» снижается, либо является нецелесообразным.

По полученным итоговым показателям и графикам, можно сделать заключение, что метод «Монте-Карло» наиболее целесообразно применять при большой объеме поступающих заявок, в то время как метод «ВМР-ранжирования» наиболее эффективен при нормированном случае поступления обращений.

Таким образом, произведен анализ реализованных методов. Выделены основные преимущества и недостатки.

3.5 Анализ разработанной системы

В рамках магистерской диссертации разработана система подачи и обработки обращений в IT-аутсорсинг. Для выделения сильных и слабых сторон разработанной системы произведен SWOT-анализ. Данный анализ позволит решить дальнейший ход развития системы, определить и устранить имеющиеся недостатки.

SWOT-анализ представляет собой метод планирования, который заключается в выявлении факторов внешней и внутренней среды проекта и разделения данных факторов на четыре категории:

- сильные стороны (Strengths);
- слабые стороны (Weaknesses);
- возможности (Opportunities);
- угрозы (Threats).

Сильные и слабые стороны являются внутренними факторами объекта анализа (на что объект способен повлиять), а возможности и угрозы относятся к факторам внешней среды (что может повлиять на объект извне).

Для проведения анализа создана и заполнена SWOT-матрица, представленная в виде таблицы 3.5.

Таблица 3.5 – Матрица SWOT-анализа

Сильные стороны	Возможности		Угрозы		Итого
	1. Расширение круга потребителей	2. Развитие системы	1. Появление конкурентов	2. Устаревание	
1.Низкая стоимость разработки	++	+	+	0	+4
2.Функциональность	++	++	+	+	+5
3. Обеспечение сопровождения	+	+	+	+	+4
Итого	+5	+4	+3	+2	+14

Продолжение таблицы 3.5

Слабые стороны	Возможности		Угрозы		Итого
	1. Расширение круга потребителей	2. Развитие системы	1. Появление конкурентов	2. Устаревание	
1. Недостаточное финансирование	-	--	0	-	-4
2. Нехватка квалифицированных кадров	-	--	0	-	-4
Итого	-2	-4	0	-2	-8
Общий итог	+3	0	+3	0	+6

Значение в ячейке равное нулю означает отсутствие влияния данного фактора, а значение в виде плюса или минуса означает, что зависимость имеется. Двойное сочетание знака плюс или минус означает, что фактор более зависим от параметра.

Проанализировав SWOT-матрицу из таблицы 3.5, можно сделать вывод, что существует трудность в выявлении самой сильной стороны, так как каждая из выделенных сильных сторон в равной степени важна. Однако наиболее важным достоинством системы является функциональность.

Отраженные в матрице слабые стороны разработки являются довольно опасными, а из рассмотренных возможностей более доступными для выполнения являются возможности расширения круга потребителей и развитие системы. Однако следует учитывать, что развитие системы из-за возможного недостатка финансирования или нехватке кадров может быть проблематичным.

Наиболее опасной угрозой является устаревание разработки. Однако данная угроза может быть своевременно устранена при развитии функционала системы.

Исходя из полученных результатов анализа, можно сделать вывод, что разработанная система является перспективной, но с незначительными слабыми сторонами, которые могут быть своевременно устранены. При преодолении

перечисленных слабостей, разработка станет более перспективной, успешной и привлекательной для потребителя. Первостепенной задачей представляется поиск путей дополнительного финансирования и дальнейшее развитие функционала на основании нужд потребителей.

Следует учитывать все неблагоприятные исходы, а так же использовать все возможности для покрытия выделенных слабых сторон проекта. В случае, если данные проблемы не будут своевременно решены, существует угроза того, что проект в будущем может стать бесперспективным.

Вывод по третьей главе

В результате третьей главы построены и отображены алгоритмы разработанной системы, реализована спроектированная база данных. На основании данных алгоритмов и базы данных разработана и протестирована информационная система подачи и обработки обращений в IT аутсорсинг.

В процессе работы проведен сравнительных анализ используемых методов распределения заявок относительно исполнителей. Приведены сравнительные показатели и графики показателей распределения.

На заключительном этапе проведен общий анализ разработанной системы путем построения матрицы для SWOT-анализа, на основании которой выделены слабые и сильные стороны проекта, пути дальнейшего развития системы и меры по устранению слабых сторон проекта и увеличению фактора воздействия сильных сторон на проект.

В результате анализа системы сделан вывод, что система является перспективной для дальнейшего развития, но с незначительными слабыми сторонами в виде недостаточного финансирования и недостатком квалифицированных кадров, своевременное решение которых позволит системе стать более перспективной и успешной для круга потребителей.

ЗАКЛЮЧЕНИЕ

В магистерской диссертации создана информационная система, реализующая функционал подачи и обработки обращений в IT аутсорсинг.

Для снижения рисков и большей продуктивности, в рамках магистерской диссертации разработана система, позволяющая указывать нужные методы выполнения работ и распределять задачи и заявки относительно предыдущих результатов работ, которые уже были успешно достигнуты инженерами.

В результате разработанной системы, была достигнута цель, которая заключалась в усовершенствовании методов подачи и обработки обращений для компаний, предоставляющих услуги IT-аутсорсинга.

Для достижения цели разработана информационная система, которая позволяет осуществлять:

– создание, контроль и мониторинг обращений в IT аутсорсинг;

– указание и изменение параметров заявок;

– обеспечение обратной связи между исполнителем и субъектом обращения;

– разграничение прав по работе с заявками заданных групп пользователей;

– более точный подбор наиболее компетентных лиц для решения заявок с помощью методов «Монте-Карло» или BPM-ранжирования.

Разработанная система является перспективной и обладает всеми возможностями и задатками для дальнейшего развития и сопровождения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Михелёв, В.М. Базы данных и СУБД: учебное пособие // В.М. Михелёв, – Б.: Издательство БелГУ, 2007. – 200 с.

2. Малыхина, М.П. Базы данных: основы, проектирование, использование // М.П. Малыхина, – СПБ.: БХВ – Петербург, 2015. – 512 с.

3. Федоров, А.Г. Базы данных для всех // А.Г. Федоров, – М.: КомпьютерПресс, 2016. – 256 с.

4. Дейт, К.П. Введение в системы баз данных // К.П. Дейт, – СПБ.: Издательский дом «Вильямс», 2015. – 848 с.

5. Стивенс, Р.О. Программирование баз данных // Р.О. Стивенс, – М.: Бином–Пресс, 2017. – 384 с.

6. Полякова, Л.Н. Основы SQL: Курс лекций. Учебное пособие. // Л.Н. Полякова, – М.: ИНТУИТ.РУ, Интернет –Университет Информационных технологий, 2016. – 368 с.

7. Карпова, Т.С. Базы данных: модели, разработка, реализация. // Т.С. Карпова, – М.: Эксмо, 2016. – 423 с.

8. Агальцов, В.П. Базы данных: Учебное пособие. / В.П. Агальцов – Москва: Мир, 2017. – 375 с.

9. Вирт, Н.А. Алгоритмы и структуры данных. // Н.А. Вирт, – СПБ.: Невский диалект, 2016. – 325 с.

10. Глушаков, С.В. Базы данных: Учебный курс. // С.В. Глушаков, – Х.: Фолио, 2016. – 504 с.

11. Самуйлов, К.Е. Бизнес-процессы и информационные технологии // К.Е. Самуйлов, – М.: Паблишер, 2014. – 446 с.

12. Вирт, Д.С. Алгоритмы и структуры данных // Д.С. Вирт, – М.: Мир, 2015. – 128 с.

13. Диденко, Н.И. Анализ и обоснование решений в маркетинге // Н.И. Диденко, Д.Ф. Скрипнюк, – М.: Высшая школа, 2015. – 304 с.

14. Маторин, С.И. Теория систем и системный анализ: учебное пособие // С.И. Маторин, О.А. Зимовец, – Б.: БУКЭП, 2016 г. – 259 с.

15. Котеров, Д.А. PHP в подлиннике // Д.А. Котеров, – СПБ.: БХВ-Петербург, 2017 г. – 1120 с.

16. Костарев, А.Ф. PHP 5 // Д.А Костарев, – СПБ.: БХВ-Петербург, 2018 г. – 1104 с.

17. Зандстра, М. PHP: объекты, шаблоны и методики программирования // М. Зандстра, – М.: Вильямс, 2018 г. – 560 с.

18. Дари, К. PHP и MySQL: создание интернет магазина // К. Дари, – М.: Вильямс, 2017 г. – 250 с.

19. Ленгсторф, Д. PHP и jQuery для профессионалов // Д. Ленгсторф, – М.: Вильямс, 2018 г. – 352 с.

20. Суэринг, С. PHP и MySQL. Библия программиста // С. Суэринг, – М.: Диалектика, 2017 г. – 912 с.

21. Зервас, К. Web 2.0: создание приложений на PHP // К. Зервас, – М.: Вильямс, 2017 г. – 544 с.

22. Кузнецов, М.А. PHP 5/6 // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2017 г. – 1024 с.

23. Кузнецов, М.А. Объектно-ориентированное программирование на PHP // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2018 г. – 608 с.

24. Леки-Томпсон, Э. PHP для профессионалов // Э. Леки-Томпсон, – М.: Диалектика, 2017 г. – 608 с.

25. Кузнецов, М.А. Самоучитель PHP 5/6 // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2018 г. – 672 с.

26. Кузнецов, М.А. PHP. Практика создания Web-сайтов // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2018 г. – 1264 с.

27. Кузнецов, М.А. Головоломки на PHP // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2018 г. – 544 с.

28. Кузнецов, М.А. PHP на примерах // М.А. Кузнецов, – СПБ.: БХВ-Петербург, 2019 г. – 400 с.

29. Аткинсон, Л. MySQL: библиотека профессионала // Л. Аткинсон, – М.: Вильямс, 2017 г. – 624 с.

30. Горев, А.В. Эффективная работа с СУБД // А.В. Горев, – СПБ.: Питер, 2018 г. – 704 с.

31. Яргер, Р.Д. MySQL и mSQL: Базы данных для небольших предприятий и интернета // Р.Д. Яргер, – СПБ.: Символ-Плюс, 2018 г. – 560 с.

32. Агафонов, А.В. Логическое программирование // А.В. Агафонов, – М.: Символ-Плюс, 2017 г. – 471 с.

33. Вайсфельд, М. Объектно-ориентированное мышление // М. Вайсфельд, – М.: Питер, 2017 г. – 624 с.

34. Иванова, Г.С. Объектно-ориентированное программирование // Г.С Иванова, – М.: Питер, 2017 г. – 361 с.

35. Кнут, Д.Э. Искусство программирования // Д.Э. Кнут, – М.: Вильямс, 2017 г. – 947 с.

36. Колесов, Ю.Б. Моделирование систем. Объектно-ориентированный подход // Ю.Б. Колесов, – М.: БХВ-Петербург, 2017 г. – 192 с.

37. Кормлев, Н.Ю. Объектно-ориентированное программирование // Н.Ю. Кормлев, – М.: Солон-Пресс, 2017 г. – 770 с.

38. Кузнецов, М.Ю. Объектно-ориентированное программирование на PHP // М.Ю. Кузнецов, – М.: БХВ-Петербург, 2017 г. – 608 с.

39. Лесневский, А.С. Объектно-ориентированное программирование // А.С. Лесневский, – М.: БХВ-Петербург, 2017 г. – 232 с.

40. Мартынов, А.А. Алгоритмизация и основы объектно-ориентированного мышления // А.А. Мартынов, – М.: Бином-Пресс, 2017 г. – 271 с.

41. Шакин, В.Н. Объектно-ориентированное программирование // В.Н. Шакин, – М.: РРГУ, 2017 г. – 118 с.

ПРИЛОЖЕНИЕ А

Код основной страницы обращений

```
<?php
/**
 * Created by PhpStorm.
 * User: User
 * Date: 004 04.01.19
 * Time: 16:09
 */
require_once("dbconnect.php"); // подключаем содержимое файла text.php
require_once("header.php");
ini_set('session.bug_compat_warn', 0);
?>
<link rel="stylesheet" type="text/css" href="css/filter_request_style.css">
<meta http-equiv="Refresh" content="30" />
<div class="block_for_messages">
    <?php
    //Если в сессии существуют сообщения об ошибках, то выводим их
    if(isset($_SESSION["error_messages"]) && !empty($_SESSION["error_messages"])){
        echo $_SESSION["error_messages"];

        //Уничтожаем чтобы не выводились заново при обновлении страницы
        unset($_SESSION["error_messages"]);
    }

    //Если в сессии существуют радостные сообщения, то выводим их
    if(isset($_SESSION["success_messages"]) && !empty($_SESSION["success_messages"])){
        echo $_SESSION["success_messages"];

        //Уничтожаем чтобы не выводились заново при обновлении страницы
        unset($_SESSION["success_messages"]);
    }
    ?>
</div>
<?php
//Проверяем, если пользователь не авторизован, то выводим форму регистрации,
//иначе выводим сообщение о том, что он уже зарегистрирован
if(isset($_SESSION["email"])        &&        isset($_SESSION["password"])        /*&&
($_SESSION['fk_Role_id'])==1*/){ //убрал "НЕ"
    if(isset($_GET['sort']) && isset($_GET['type']))
    {
        $_SESSION['order_request_column_name'] = $_GET['sort'];
        $_SESSION['order_request_type'] = $_GET['type'];
    }
    if (isset($_GET['page'])) {
        $_SESSION['current_page'] = $_GET['page'];
    }
    ?>
    <br><br>
     <?php //ПРЕФИЛЬТР
    $query="
select req.request_id as request_id,
req.caption as caption,
req.short_description as short_description,
req.date_create as date_create,
IFNULL(req.date_resolve, 'Не решена') as date_resolve,
concat(userCreate.last_name,         \"         \",userCreate.first_name,\"         \",
\"(\",userCreate.email,\")\") as userCreate,
```

```
userCreate.user_id as userCreateID,
concat(userRespons.last_name,         \"         \",userRespons.first_name,\"         \",
\"(\",userRespons.email,\")\") as userRespons,
userRespons.user_id as userResponsID,
org.organisation_name as organisation_name,
org.id_organisation as id_organisation,
status.status_name as status_name,
status.status_id as status_id,
priority.priority_name as priority_name,
priority.priority_id as priority_id,
service.service_name as service_name,
service.service_id as service_id
from requests req
inner join users userCreate on req.fk_create_user_id = userCreate.user_id
inner join users userRespons on req.fk_responsible_user_id = userRespons.user_id
inner join organisations org on org.id_organisation = userCreate.fk_organisation_id
inner join status status on req.fk_status_id = status.status_id
inner join priority priority on priority.priority_id = req.fk_priority_id
inner join service service on req.fk_service_id = service.service_id
where request_id > 0";

    //Префильтр дл конкретного типа роли пользователя
    $prefilter = '';
    switch ($_SESSION['fk_Role_id'])
    {
        case 1: $prefilter = ' '; break;
        case 2: $prefilter = ' and userRespons.user_id = '.$_SESSION['user_id']; break;
        case 3: $prefilter = ' and userCreate.user_id = '.$_SESSION['user_id']; break;
        default: $prefilter = ' '; break;
    }
    $prefilter_organisation = '';

    if($_SESSION['fk_Role_id'] != 1)
    {
        $query_org_curr_user = $mysqli->query("SELECT fk_organisation_id FROM `users`
WHERE user_id = '" . $_SESSION['user_id'] . "'");
        $row_org_curr_user = mysqli_fetch_assoc($query_org_curr_user);

        $prefilter_organisation .= ' and ' . $row_org_curr_user['fk_organisation_id'] .
' in (userCreate.fk_organisation_id,userRespons.fk_organisation_id) ';
    }
    $prefilter.=$prefilter_organisation;
    //Фильтр для выбранных значений
    $filter_request_id= '';
    if(isset($_POST['filter_request_id']) and $_POST['filter_request_id']!='') {
        $filter_request_id =' And req.request_id='.$_POST['filter_request_id'];
        $_SESSION['filter_request_id'] = $_POST['filter_request_id'];
    }
    else if ($_POST['filter_request_id']=='')
    {
        $_SESSION['filter_request_id'] = null;
    }
    //Заголовок обращения
    $filter_caption= '';
    if(isset($_POST['filter_caption']) and $_POST['filter_caption']!='') {
        $filter_caption =' And req.caption like \'%'.$_POST['filter_caption'].'%\'';
        $_SESSION['filter_caption_request'] = $_POST['filter_caption'];
    }
    else if($_POST['filter_caption']=='')
    {
        $_SESSION['filter_caption_request'] = null;
    }
```

```
    //Краткое описание filter_short_description
    $filter_short_description = '';
    if(isset($_POST['filter_short_description'])                                    and
$_POST['filter_short_description']!='') {
        $filter_caption        ='        And        req.short_description        like
\'%'.$_POST['filter_short_description'].'%\'';
        $_SESSION['filter_short_description_request']                                =
$_POST['filter_short_description'];
    }
    else if($_POST['filter_short_description']=='')
    {
        $_SESSION['filter_short_description_request'] = null;
    }
    //Создатель обращения
     if(isset($_POST['option1']) and $_POST['option1']!='' and $_POST['option1']!='Все
пользователи') {

         $userCreate = trim($_POST["option1"]);
         $userCreate = htmlspecialchars($userCreate, ENT_QUOTES);

         //поиск сначала значений в скобках, потом берем email регулярками
         $userCreate = preg_match_all("/\(([^()]*)\)/", $userCreate, $matches);
         $userCreate = $matches[0][0];
         $userCreate      =      preg_match_all("/[\._a-zA-Z0-9-]+@[\._a-zA-Z0-9-]+/i",
$userCreate, $matches2);
         $userCreate = $matches2[0][0];

         $result_query_userCreate = $mysqli->query("select user_id,concat(last_name, \"
\",first_name,\" \", \"(\",email,\")\") as userCreate from users where user_id >0
and fk_role_id =3 and email='" . $userCreate . "'");
         $userCreateID = mysqli_fetch_assoc($result_query_userCreate);
         $resultUserCreateID = $userCreateID['user_id'];

         $filter_user_create_id= ' And req.fk_create_user_id='.$resultUserCreateID;

         $_SESSION['filter_user_create_request'] = $userCreateID['user_id'];
     }
     else if ($_POST['option1']=='Все пользователи')
     {
         $_SESSION['filter_user_create_request'] = null;
     }

    //Исполнитель обращения
    if(isset($_POST['option2']) and $_POST['option2']!='' and $_POST['option2']!='Все
исполнители') {

        $userRespons = trim($_POST["option2"]);
        $userRespons = htmlspecialchars($userRespons, ENT_QUOTES);

        //поиск сначала значений в скобках, потом берем email регулярками
        $userRespons = preg_match_all("/\(([^()]*)\)/", $userRespons, $matches);
        $userRespons = $matches[0][0];
        $userRespons      =      preg_match_all("/[\._a-zA-Z0-9-]+@[\._a-zA-Z0-9-]+/i",
$userRespons, $matches2);
        $userRespons = $matches2[0][0];

        $result_query_userRespons = $mysqli->query("select user_id,concat(last_name, \"
\",first_name,\" \", \"(\",email,\")\") as userRespons from users where user_id >0
and fk_role_id =2 and email='" . $userRespons . "'");
        $userResponsID = mysqli_fetch_assoc($result_query_userRespons);
        $resultUserResponsID = $userResponsID['user_id'];
```

```
        $filter_user_respons_id=                                ' And
req.fk_responsible_user_id='.$resultUserResponsID;

        $_SESSION['filter_user_response_request'] = $userResponsID['user_id'];
    }
    else if ($_POST['option2']=='Все исполнители')
    {
        $_SESSION['filter_user_response_request'] = null;
    }

    //Дата создания от
    $filter_date_create_begin = '';
    if(isset($_POST['filter_date_create_begin'])                                    and
$_POST['filter_date_create_begin']!=''                                              and
$_POST['filter_date_create_begin']!='дд.мм.гггг') {
        $filter_date_create_begin           ='           And           req.date_create
>=\''.$_POST['filter_date_create_begin'].'T00:00:00\' ';

        $_SESSION['filter_date_create_begin_request']                                =
$_POST['filter_date_create_begin'];
    }
    else            if            ($_POST['filter_date_create_begin']==''           ||
$_POST['filter_date_create_begin']=='дд.мм.гггг')
    {
        $_SESSION['filter_date_create_begin_request'] = null;
    }

    //Дата создания до
    $filter_date_create_end = '';
    if(isset($_POST['filter_date_create_end']) and $_POST['filter_date_create_end']!=''
and $_POST['filter_date_create_end']!='дд.мм.гггг') {
        $filter_date_create_end           ='           And           req.date_create
<=\''.$_POST['filter_date_create_end'].'T23:59:59\' ';

        $_SESSION['filter_date_create_end_request'] = $_POST['filter_date_create_end'];
    }
    else          if          (          $_POST['filter_date_create_end']==''          ||
$_POST['filter_date_create_end']=='дд.мм.гггг')
    {
        $_SESSION['filter_date_create_end_request'] = null;
    }

    //Дата решения от
    $filter_date_resolve_begin = '';
    if(isset($_POST['filter_date_resolve_begin'])                                   and
$_POST['filter_date_resolve_begin']!=''                                             and
$_POST['filter_date_resolve_begin']!='дд.мм.гггг') {
        $filter_date_resolve_begin          ='          And          req.date_resolve
>=\''.$_POST['filter_date_resolve_begin'].'T00:00:00\' ';

        $_SESSION['filter_date_resolve_begin_request']                               =
$_POST['filter_date_resolve_begin'];
    }
    else            if           ($_POST['filter_date_resolve_begin']==''           ||
$_POST['filter_date_resolve_begin']=='дд.мм.гггг')
    {
        $_SESSION['filter_date_resolve_begin_request'] = null;
    }
    //Дата решения до
    $filter_date_resolve_end = '';
```

```
    if(isset($_POST['filter_date_resolve_end'])                                    and
$_POST['filter_date_resolve_end']!=''                                              and
$_POST['filter_date_resolve_end']!='дд.мм.гггг') {
        $filter_date_resolve_end           ='           And           req.date_resolve
<=\''.$_POST['filter_date_resolve_end'].'T23:59:59\' ';

        $_SESSION['filter_date_resolve_end_request']                                =
$_POST['filter_date_resolve_end'];
    }
    else            if            ($_POST['filter_date_resolve_end']==''            ||
$_POST['filter_date_resolve_end']=='дд.мм.гггг')
    {
        $_SESSION['filter_date_resolve_end_request'] = null;
    }

    //Статус
     if(isset($_POST['status']) and $_POST['status']!='') {

         $selected_status = $_POST['status'];
         $filter_status_id = ' And req.fk_status_id in(';
         foreach($selected_status AS $key=>$values)
         {
             if($values!= 'Все статусы')
             {
                 $filtr_statusname = $arr;
                 $filtr_statusname = htmlspecialchars($filtr_statusname, ENT_QUOTES);
                 $result_query_Status_id = $mysqli->query("select status_id,status_name
from status where status_iD >0 and status_name='" . $values . "'");
                 $statusid = mysqli_fetch_assoc($result_query_Status_id);
                 $resultStatusID = $statusid['status_id'];
                 $filter_status_id .= $resultStatusID.',';
             }
             else {
                 $result_query_Status_id = $mysqli->query("select status_id,status_name
from status where status_iD >0");
                 $statusid = mysqli_fetch_assoc($result_query_Status_id);
                 $resultStatusID = $statusid['status_id'];
                 $filter_status_id .= $resultStatusID.',';


                 $result_query_status  =  $mysqli->query("select  status_id,status_name
from status where status_iD >0");
                 $result_query_num_status = mysqli_num_rows($result_query_status);
                 for ($i=0; $i <$result_query_num_status; $i++)
                 {  /*Сделать  проверку  на  текущий  статус  в  заявке  и  выводимыми
статусами*/
                     $result = mysqli_fetch_array($result_query_status);
                     $filter_status_id.=$result['status_id'].',';
                 }
                 break;
             }
         }
         $filter_status_id = trim($filter_status_id, ',');
         $filter_status_id .= ')';
     }

    //Приоритет
    if(isset($_POST['priority']) and $_POST['priority']!='') {

        $selected_priority = $_POST['priority'];
        $filter_priority_id = ' And req.fk_priority_id in(';
        foreach($selected_priority AS $key=>$values)
```

```
        {
            if($values!= 'Все приоритеты')
            {
                $filtr_priorityname = $arr;
                $filtr_priorityname       =       htmlspecialchars($filtr_priorityname,
ENT_QUOTES);
                $result_query_Priority_id            =            $mysqli->query("select
priority_id,priority_name  from  priority  where  priority_id  >0  and  priority_name='"  .
$values . "'");
                $priorityid = mysqli_fetch_assoc($result_query_Priority_id);
                $resultPriorityID = $priorityid['priority_id'];
                $filter_priority_id .= $resultPriorityID.',';
            }
            else {
                $result_query_Priority_id            =            $mysqli->query("select
priority_id,priority_name from priority where priority_id >0");
                $priorityid = mysqli_fetch_assoc($result_query_Priority_id);
                $resultPriorityID = $priorityid['priority_id'];
                $filter_priority_id .= $resultPriorityID.',';


                $result_query_priority             =             $mysqli->query("select
priority_id,priority_name from priority where priority_id >0");
                $result_query_num_priority = mysqli_num_rows($result_query_priority);
                for ($i=0; $i <$result_query_num_priority; $i++)
                {  /*Сделать  проверку  на  текущий  статус  в  заявке  и  выводимыми
статусами*/
                    $result = mysqli_fetch_array($result_query_priority);
                    $filter_priority_id.=$result['priority_id'].',';
                }
                break;
            }
        }
        $filter_priority_id = trim($filter_priority_id, ',');
        $filter_priority_id .= ')';
    }

    //Тип услуги
    if(isset($_POST['service']) and $_POST['service']!='') {

        $selected_service = $_POST['service'];
        $filter_service_id = ' And req.fk_service_id in(';
        foreach($selected_service AS $key=>$values)
        {
            if($values!= 'Все типы услуг')
            {
                $filtr_servicename = $arr;
                $filtr_servicename = htmlspecialchars($filtr_servicename, ENT_QUOTES);
                $result_query_Service_id            =            $mysqli->query("select
service_id,service_name,fk_service_id   from   service   where   service_id   >0   and
service_name='" . $values . "'");
                $serviceid = mysqli_fetch_assoc($result_query_Service_id);
                $resultServiceID = $serviceid['service_id'];
                $filter_service_id .= $resultServiceID.',';
            }
            else {
                $result_query_Service_id            =            $mysqli->query("select
service_id,service_name,fk_service_id from service where service_id >0");
                $serviceid = mysqli_fetch_assoc($result_query_Service_id);
                $resultServiceID = $serviceid['service_id'];
                $filter_service_id .= $resultServiceID.',';
```

```
                $result_query_service              =              $mysqli->query("select
service_id,service_name,fk_service_id from service where service_id >0");
                $result_query_num_service = mysqli_num_rows($result_query_service);
                for ($i=0; $i <$result_query_num_service; $i++)
                {  /*Сделать  проверку  на  текущий  статус  в  заявке  и  выводимыми
статусами*/
                    $result = mysqli_fetch_array($result_query_service);
                    $filter_service_id.=$result['service_id'].',';
                }
                break;
            }
        }
        $filter_service_id = trim($filter_service_id, ',');
        $filter_service_id .= ')';
    }

    //Префильтр для OrderBy
    $orderBy = '';
    $orderBy     =     '     order     by     '.$_SESSION['order_request_column_name'].'
'.$_SESSION['order_request_type'];

    //Проставляем условия префильтра
    $query.=$prefilter;

    if (isset($_POST["filter_reset"]) && !empty($_POST["filter_reset"]))
    {
        $_SESSION['filter_requests'] = '';
    }

    //Проверяем были ли нажаты кнопки фильтрации или сброса фильтра
     if (isset($_POST["filter_submit"]) && !empty($_POST["filter_submit"])) {
         $_SESSION['filter_requests'] = '';
         //Проставляем условия выбранных фильтров и очищаем
         $filtr .= $filter_request_id; $filter_request_id= '';
         $filtr .= $filter_caption; $filter_caption= '';
         $filtr .= $filter_short_description; $filter_short_description= '';
         $filtr .= $filter_user_create_id; $filter_user_create_id= '';
         $filtr .= $filter_user_respons_id; $filter_user_respons_id= '';
         $filtr .= $filter_date_create_begin; $filter_date_create_begin= '';
         $filtr .= $filter_date_create_end; $filter_date_create_end= '';
         $filtr .= $filter_date_resolve_begin; $filter_date_resolve_begin= '';
         $filtr .= $filter_date_resolve_end; $filter_date_resolve_end= '';
         $filtr .= $filter_status_id; $filter_status_id= '';
         $filtr .= $filter_priority_id; $filter_priority_id= '';
         $filtr .= $filter_service_id; $filter_service_id= '';
         $_SESSION['filter_requests'].= $filtr;
     }

    $_SESSION['order_requests']='';
    $_SESSION['order_requests'].=$orderBy;
    $query.=$_SESSION['filter_requests'];
    ?>
    <!-- Скрипт для скрывающей панели -->
    <script type="text/javascript">
        $(document).ready(function(){
            $('.spoiler_links').click(function(){
                $(this).parent().children('div.spoiler_body').toggle('normal');
                return false;
            });
        });
    </script>
```

```
    <!-- ФОРМА ФИЛЬТРАЦИИ ОБРАЩЕНИЙ  -->
    <div>
        <br>
        <a class="spoiler_links" style="vertical-align:middle">Фильтр</a><br>
        <div class="spoiler_body">
            <form action="form_request.php" method="post" id="form_request">

                <label class = "HeadersLabel">Фильтр по заголовкам</label><br><br>
                <div class="input">
                    <div class="pole">
                        <div class="polesmall">
                        <label>Номер:</label>
                        <div  class="input"><input  type="text"  id="filter_request_id"
name="filter_request_id" value="<?php echo $_SESSION['filter_request_id'] ?>"/></div>
                        </div>

                        <div class="polemedium">
                        <label>Заголовок:</label>
                        <div   class="input"><input   type="text"   id="filter_caption"
name="filter_caption" value="<?php echo $_SESSION['filter_caption_request'] ?>"/></div>

                        <label>Краткое описание:</label>
                        <div              class="input"><input              type="text"
id="filter_short_description"    name="filter_short_description"    value="<?php    echo
$_SESSION['filter_short_description_request'] ?>"/></div>
                        </div>
                    </div>

                    <hr><label       class       =       "HeadersLabel">Фильтр       по
пользователям</label></hr><br><br>
                    <div class="poleUsers">
                        <label>Создатель:</label>
                        <select name="option1" class="cellbut">
                            <?php
                           if($_SESSION['filter_user_create_request'] == null)
                            {
                                echo "<option selected >Все пользователи</option>";
                            }
                            else
                            {
                                echo "<option>Все пользователи</option>";
                            }
                            ?>

                            <?php  //Option  для  выбора  фильтра  по  создавшему  заявку
пользователю

                            $result_query_users         =         $mysqli->query("select
user_id,concat(last_name, \" \",first_name,\" \", \"(\",email,\")\") as userCreate from
users where user_id >0 and fk_role_id =3");
                            $result_query_num_users                                      =
mysqli_num_rows($result_query_users);
                            for ($i=0; $i <$result_query_num_users; $i++)
                            {
                                $result = mysqli_fetch_array($result_query_users);
                                if($_SESSION['filter_user_create_request']           ==
$result['user_id'])
                                {
                                    echo '<option selected> ' . $result['userCreate'] .
'</option>';
                                }
                                else {
```

```
                                    echo  '<option  >  '  .  $result['userCreate']  .
'</option>';
                                }

                            }
                            ?>
                        </select>
                        <label>Исполнитель:</label>
                        <select name="option2" class="cellbut">
                           <?php
                            if($_SESSION['filter_user_response_request'] == null)
                            {
                                echo "<option selected >Все исполнители</option>";
                            }
                            else
                            {
                                echo "<option>Все исполнители</option>";
                            }
                            ?>
                            <?php  //Option  для  выбора  фильтра  по  создавшему  заявку
пользователю

                            $result_query_users         =         $mysqli->query("select
user_id,concat(last_name,  \"  \",first_name,\"  \",  \"(\",email,\")\")  as  userRespons
from users where user_id >0 and fk_role_id =2");
                            $result_query_num_users                                    =
mysqli_num_rows($result_query_users);
                            for ($i=0; $i <$result_query_num_users; $i++)
                            {
                                $result = mysqli_fetch_array($result_query_users);

                                if($_SESSION['filter_user_response_request']         ==
$result['user_id'])
                                {
                                    echo  '<option  selected>  '  .  $result['userRespons']
. '</option>';
                                }
                                else {
                                    echo  '<option  >  '  .  $result['userRespons']  .
'</option>';
                                }

                            }
                            ?>
                        </select>
                    </div>
                    <hr><label       class       =       "HeadersLabel">Фильтр       по
датам</label></hr><br><br>
                    <div class="poledateCreate">
                        <label>Дата создания от:</label>
                        <div              class="input"><input              type="date"
id="filter_date_create_begin"    name="filter_date_create_begin"    value="<?php    echo
$_SESSION['filter_date_create_begin_request'] ?>"/></div>
                        <label>Дата создания до:</label>
                        <div              class="input"><input              type="date"
id="filter_date_create_end"     name="filter_date_create_end"     value="<?php     echo
$_SESSION['filter_date_create_end_request'] ?>"/></div>
                    </div>
                    <div class="poledateEnd">
                        <label>Дата решения от:</label>
```

```
                        <div              class="input"><input              type="date"
id="filter_date_resolve_begin"   name="filter_date_resolve_begin"   value="<?php   echo
$_SESSION['filter_date_resolve_begin_request'] ?>"/></div>
                        <label>Дата решения до:</label>
                        <div              class="input"><input              type="date"
id="filter_date_resolve_end"    name="filter_date_resolve_end"    value="<?php    echo
$_SESSION['filter_date_resolve_end_request'] ?>"/></div>

                    </div>

                    <hr><label   class   =   "HeadersLabel">Фильтр   по   дополнительным
данным</label></hr><br><br>
                    <div class="poleSelect">

                        <div class="polemultiselectSmall">
                        <label>Статус:</label>
                        <select name="status[]" class="cellbut" multiple="yes" >
                        <option selected >Все статусы</option>
                        <?php //Option для выбора статуса
                        $result_query_status          =          $mysqli->query("select
status_id,status_name from status where status_iD >0");
                        $result_query_num_status                                        =
mysqli_num_rows($result_query_status);
                        for ($i=0; $i <$result_query_num_status; $i++)
                        { /*Сделать проверку на текущий статус в заявке и выводимыми
статусами*/
                            $result = mysqli_fetch_array($result_query_status);
                                echo '<option> '.$result['status_name'].'</option>';
                        }
                        ?>
                        </select>

                        <label>Приоритет:</label>
                        <select name="priority[]" class="cellbut" multiple="yes">
                        <option selected >Все приоритеты</option>
                        <?php //Option для выбора приоритета
                        $result_query_priority         =         $mysqli->query("select
priority_id,priority_name from priority where priority_id >0");
                        $result_query_num_priority                                      =
mysqli_num_rows($result_query_priority);
                        for ($i=0; $i <$result_query_num_priority; $i++)
                        { /*Сделать проверку на текущий приоритет в заявке и выводимыми
приоритетами*/
                            $result = mysqli_fetch_array($result_query_priority);
                            echo '<option> '.$result['priority_name'].'</option>';
                        }
                        ?>
                        </select>
                        </div>

                        <div class="polemultiselect">
                        <label>Тип проблемы / услуги:</label>
                        <select name="service[]" class="cellbut" multiple="yes">
                        <option selected >Все типы услуг</option>
                        <!--<select name="option4" class="cellbut">-->
                        <?php //Option для выбора услуги / проблемы
                        $result                =                $mysqli->query("select
service_id,service_name,fk_service_id from service where service_id >0");

                        $cats = array(); // тут будет наш массив с категориями каталога
                        // в цикле формируем нужный нам массив
                        while($cat =  mysqli_fetch_assoc($result))
```

```
                        $cats[$cat['fk_service_id']][] =  $cat;
                    // далее наша главная, рекурсивная функция, которая сформирует
дерево категорий

                    function create_tree ($cats,$fk_service_id , $row_request){
                        if(is_array($cats) and  isset($cats[$fk_service_id])){
                            $tree = '';
                            foreach($cats[$fk_service_id] as $cat){
                                $tree .= "<option>".$cat['service_name'];

                                $tree .= '</option>';
                                $tree  .=    create_tree  ($cats,$cat['service_id'],
$row_request);

                            }
                        }
                        else return null;
                        return $tree;
                    }

                    // вызываем функцию и строим дерево
                    echo create_tree ($cats, 0, $row_request['service_id']);

                    ?>
                    </select>
                    </div>
                </div>
            </div>
            <div class="sub">
                <input name="filter_submit" type="submit" value="Отфильтровать"/>
                <input name="filter_reset" type="submit" value="Сбросить фильтр"/>
            </div>

        </form>
    </div>
</div>
<br>
<!--/*Старт формы обращений*/-->
<table>
    <thead><!-- необязательный тег-->
    <tr>
        <th class="request_id" id="request_id">№<br>
            <a              href='./form_request.php?sort=request_id&type=asc'><img
src="./background/sort_asc.png"></a>
                 <a    href='./form_request.php?sort=request_id&type=desc'><img
src="./background/sort_desc.png"></a>
        </th>
        <th class="caption" id="caption">Заголовок<br>
            <a                 href='./form_request.php?sort=caption&type=asc'><img
src="./background/sort_asc.png"></a>
                  <a     href='./form_request.php?sort=caption&type=desc'><img
src="./background/sort_desc.png"></a>
        </th>
        <th class="short_description" id="short_description">Краткое описание<br>
            <a        href='./form_request.php?sort=short_description&type=asc'><img
src="./background/sort_asc.png"></a>
                                                                               <a
href='./form_request.php?sort=short_description&type=desc'><img
src="./background/sort_desc.png"></a>
        </th>
        <th class="date_create" id="date_create">Дата создания<br>
```

```
                <a              href='./form_request.php?sort=date_create&type=asc'><img
src="./background/sort_asc.png"></a>
                     <a    href='./form_request.php?sort=date_create&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="date_resolve" id="date_resolve">Дата решения<br>
                <a             href='./form_request.php?sort=date_resolve&type=asc'><img
src="./background/sort_asc.png"></a>
                    <a   href='./form_request.php?sort=date_resolve&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="userCreate" id="userCreate">Обратился<br>
                <a        href='./form_request.php?sort=fk_create_user_id&type=asc'><img
src="./background/sort_asc.png"></a>
                                                                                   <a
href='./form_request.php?sort=fk_create_user_id&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="userRespons" id="userRespons">Исполнитель<br>
                <a   href='./form_request.php?sort=fk_responsible_user_id&type=asc'><img
src="./background/sort_asc.png"></a>
                                                                                   <a
href='./form_request.php?sort=fk_responsible_user_id&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            </th>
            <th      class="organisation_name"      id="organisation_name">Обратившаяся
организация<br>
                <a          href='./form_request.php?sort=id_organisation&type=asc'><img
src="./background/sort_asc.png"></a>
                  <a href='./form_request.php?sort=id_organisation&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="status_name" id="status_name">Статус<br>
                <a             href='./form_request.php?sort=fk_status_id&type=asc'><img
src="./background/sort_asc.png"></a>
                    <a   href='./form_request.php?sort=fk_status_id&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="priority_name" id="priority_name">Приоритет<br>
                <a           href='./form_request.php?sort=fk_priority_id&type=asc'><img
src="./background/sort_asc.png"></a>
                   <a  href='./form_request.php?sort=fk_priority_id&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
            <th class="service_name" id="service_name">Тип услуги<br>
                <a              href='./form_request.php?sort=service_id&type=asc'><img
src="./background/sort_asc.png"></a>
                     <a    href='./form_request.php?sort=service_id&type=desc'><img
src="./background/sort_desc.png"></a>
            </th>
        </tr>
        </thead>
        <tbody><!--необязательный тег-->
    <?php
    // Устанавливаем количество записей, которые будут выводиться на одной странице
    // Поставьте нужное вам число. Для примера я указал одну запись на страницу
    $quantity=30;

    // Ограничиваем количество ссылок, которые будут выводиться перед и
    // после текущей страницы
    $limit=3;
```

```
    // Если значение page= не является числом, то показываем
    // пользователю первую страницу
    if(!is_numeric($page)) $page=1;

    // Если пользователь вручную поменяет в адресной строке значение page= на нуль,
    // то мы определим это и поменяем на единицу, то-есть отправим на первую
    // страницу, чтобы избежать ошибки
    if ($page<1) $page=1;

    // Узнаем количество всех доступных записей
    $q="select count(*)
from requests req
inner join users userCreate on req.fk_create_user_id = userCreate.user_id
inner join users userRespons on req.fk_responsible_user_id = userRespons.user_id
inner join organisations org on org.id_organisation = userCreate.fk_organisation_id
inner join status status on req.fk_status_id = status.status_id
inner join priority priority on priority.priority_id = req.fk_priority_id
inner join service service on req.fk_service_id = service.service_id
where request_id > 0 ".$prefilter;
    $q.=$_SESSION['filter_requests'];
    $q.=$_SESSION['order_requests'];


    // получаем номер страницы
    if (isset($_GET['page'])) $page=($_GET['page']-1); else $page=0;
    // вычисляем первый оператор для LIMIT

    $result2=$mysqli->query($q);
    $num_r = mysqli_fetch_row($result2);
    $num = $num_r[0];
    // Вычисляем количество страниц, чтобы знать сколько ссылок выводить
    $pages = $num/$quantity;

    // Округляем полученное число страниц в большую сторону
    $pages = ceil($pages);

    // Здесь мы увеличиваем число страниц на единицу чтобы начальное значение было
    // равно единице, а не нулю. Значение page= будет
    // совпадать с цифрой в ссылке, которую будут видеть посетители
    $pages++;

    // Если значение page= больше числа страниц, то выводим первую страницу
    if ($page>$pages) $page = 1;
    // Переменная $list указывает с какой записи начинать выводить данные.
    // Если это число не определено, то будем выводить
    // с самого начала, то-есть с нулевой записи
    if (!isset($list)) $list=0;

    // Чтобы у нас значение page= в адресе ссылки совпадало с номером
    // страницы мы будем его увеличивать на единицу при выводе ссылок, а
    // здесь наоборот уменьшаем чтобы ничего не нарушить.
    $list=$page*$quantity;

    $orderby_pagination = $orderBy.' LIMIT '.$quantity.' OFFSET '. $list;

    $query .= $orderby_pagination;
    $result_query_requests = $mysqli->query($query);
    $result_query_num_requests = mysqli_num_rows($result_query_requests);
    for ($i=0; $i <$result_query_num_requests; $i++)
    {
        $result = mysqli_fetch_array($result_query_requests);
```

```
    ?>
    <tr       onclick="window.location='./form_edit_request.php?request=<?php       echo
$result['request_id'] ?>'"> <!--Передаем ID обращения-->
        <td class="request_id">
            <?php echo $result['request_id']?>
        </td>

        <td class="caption">
            <?php echo $result['caption']?>
        </td>

        <td class="short_description">
            <?php echo $result['short_description']?>
        </td>

        <td class="date_create">
            <?php echo date("d.m.Y H:i", strtotime($result['date_create']))?>
        </td>

        <td class="date_resolve">
            <?php
            if($result['date_resolve'] == 'Не решена' || $result['date_resolve'] ==
'0000-00-00 00:00:00')
            {
                echo $result['date_resolve'];
            }
            else
                {
                echo date("d.m.Y H:i", strtotime($result['date_resolve']));
                }
            ?>
        </td>

        <td class="userCreate">
           <?php echo $result['userCreate']?>
        </td>

        <td class="userRespons">
            <?php echo $result['userRespons']?>
        </td>
        <td class="organisation_name">
           <?php echo $result['organisation_name']?>
        </td>

        <td class="status_name">
            <?php echo $result['status_name']?>
        </td class="request_id">

        <td class="priority_name">
            <?php echo $result['priority_name']?>
        </td>

        <td class="service_name">
            <?php echo $result['service_name']?>
        </td>
    </tr>
    <?php
    }
    ?>
        </tbody>
    </table>
    <!--Конец формы обращений -->
```

```
    <?php
    // дальше выводим ссылки на страницы:
    echo 'Страницы: ';

// _____________________ начало блока 1 _____________________

// Выводим ссылки "назад" и "на первую страницу"
    if ($page>=1) {

        // Значение page= для первой страницы всегда равно единице,
        // поэтому так и пишем
        echo        '<a        href="'        .        $_SERVER['SCRIPT_NAME']        .
'?page=1&sort='.$_SESSION['order_request_column_name'].'&type='.$_SESSION['order_reques
t_type'].'"><<</a>   ';

        // Так как мы количество страниц до этого уменьшили на единицу,
        // то для того, чтобы попасть на предыдущую страницу,
        // нам не нужно ничего вычислять
        echo    '<a    href="'    .    $_SERVER['SCRIPT_NAME']    .    '?page='    .    $page
.'&sort='.$_SESSION['order_request_column_name'].'&type='.$_SESSION['order_request_type
'].
            '"><  </a>   ';
    }

// _____________________ конец блока 1 _____________________

// На данном этапе номер текущей страницы = $page+1
    if (isset($_GET['page'])) $_this=($_GET['page']+1); else $_this=0;

// Узнаем с какой ссылки начинать вывод
    $start = $_this-$limit;

// Узнаем номер последней ссылки для вывода
    $end = $_this+$limit;

// Выводим ссылки на все страницы
// Начальное число $j в нашем случае должно равнятся единице, а не нулю
    for ($j = 1; $j<$pages; $j++) {

        // Выводим ссылки только в том случае, если их номер больше или равен
        // начальному значению, и меньше или равен конечному значению
        if ($j>=$start && $j<=$end) {

            // Ссылка на текущую страницу выделяется жирным
            if ($j==($page+1)) echo '<a href="' . $_SERVER['SCRIPT_NAME'] .
                '?page=' . $j . '"><strong style="color: #df0000">' . $j .
                '</strong></a>   ';

            // Ссылки на остальные страницы
            else echo '<a href="' . $_SERVER['SCRIPT_NAME'] . '?page=' .
                $j . '">' . $j . '</a>   ';
        }
    }

// _____________________ начало блока 2 _____________________

// Выводим ссылки "вперед" и "на последнюю страницу"
    if ($j>$page && ($page+2)<$j) {

        // Чтобы попасть на следующую страницу нужно увеличить $pages на 2
        echo '<a href="' . $_SERVER['SCRIPT_NAME'] . '?page=' . ($page+2) .
            '"> ></a>   ';
```

```
        // Так как у нас $j = количество страниц + 1, то теперь
        // уменьшаем его на единицу и получаем ссылку на последнюю страницу
        echo '<a href="' . $_SERVER['SCRIPT_NAME'] . '?page=' . ($j-1) .
            '">>></a>   ';
    }

// _____________________ конец блока 2 _____________________

    $mysqli->close();
    ?>
    <?php
}else{
    ?>
    <div id="authorized">
        <br><br><h1>Для просмотра обращений, требуется авторизация</h1>
    </div>
    <?php
}

//Подключение подвала
require_once("footer.php");
?>
```

Магистерская диссертационная работа выполнена мной совершенно самостоятельно. Все использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

«___» _____ _____ г.

_____ _____

(подпись) *(Ф.И.О.)*