Министерство науки и высшего образования Российской Федерации
Филиал Федерального государственного автономного образовательного учреждения
высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
в г. Нижневартовске

Кафедра «Гуманитарные, естественно-научные и технические дисциплины»

ДОПУСТИТЬ К ЗАЩИТЕ

Зав. кафедрой «ГЕНТД»
к.филос.н., доцент
_____/И.Г. Рябова /
«____»_____ 2021 г.

Разработка информационной системы учета телефонных разговоров сотрудников

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ-09.03.04.2021.040.ПЗ ВКР

Консультанты
Экономическая часть
к.э.н., доцент
_____/С.В.Данилова/
«_____»_____ 2021 г.

Руководитель работы
исполнительный директор
ООО «Нефтегазтехника»
_____/ Ф.Х. Азизов /
«_____»_____ 2021 г.

Автор работы
Обучающийся группы НвФл-422
_____/ Б.В. Дмитриева /
«_____»_____ 2021 г.

Нормоконтролер
старший преподаватель
_____/ Л.Н. Буйлушкина /
«_____»_____ 2021 г.

Нижневартовск, 2021

АННОТАЦИЯ

Дмитриева Б.В. Разработка информационной системы учета телефонных разговоров сотрудников – Нижневартовск: филиал ЮУрГУ, НвФл-422: 2021, 102 стр., 25 ил., 14 табл., библиогр. список – 32 наим., 2 прил.

Данная выпускная квалификационная работа является технической и представляет собой описание разработки информационной системы учета телефонных разговоров сотрудников для ИП Земан П. С.

Предоставлена характеристика объекта автоматизации ИП. Изучена предметная область и бизнес-процессы. В данной работе описана разработка программного обеспечения, которое учитывает телефонные переговоры и формирует соответствующие отчеты. Выполнен расчет технико-экономической эффективности от внедрения информационной системы учета телефонных разговоров сотрудников. Проведен литературный обзор.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность описываемой выпускной квалификационной работы состоит в том, что главным направлением обновления системы управления любой компании в настоящее время является использование современных телекоммуникационных и IT технологий, а также реализация на их базе результативных информационно-управленческих методик.

Использование таких систем дает повышенную эффективность принятия решений, помогает объединять информационные процессы, оптимизировать организацию документооборота компании, удалить повторяющиеся функций, повысить эффективность деятельности в целом, минимизировать расходы на информационное сопровождение работы фирмы.

В настоящее время фирмам достаточно сложно удержать лидирующие позиции, в связи с конкурентоспособностью. Обуславливая продвижение прогрессивных IT-технологий в производстве. Обобщение методов управления затратами, возможное за счет применения автоматизированной информационной системы(далее – АИС), ведет к тому, что уже не первый год конкурентные преимущества ищут в оптимизации работы.

В настоящей выпускной квалификационной работе рассматривается вопрос разработки информационной системы учета телефонных разговоров сотрудников.

Объектом исследования является деятельность ресторана доставки «Суши АНТАНО» ИП Земан П.С.

Предмет исследования – автоматизация учета телефонных разговоров сотрудников, работающих в ИП.

Цель работы – разработать информационную систему учета телефонных разговоров сотрудников

Задачи работы:

– анализ предметной области внедрения автоматизируемой информационной системы;

– обзор и выбор средств разработки информационной системы;

– разработка технического задания;

– разработка функциональной схемы и базы данных для информационной системы;

– описание разработанного программного продукта;

– оценка экономической эффективности информационной системы учета телефонных разговоров сотрудников.

В качестве средства достижения поставленной цели будет использоваться моделирование бизнес-процессов и разработка порядка их автоматизации.

Выпускная квалификационная работа состоит из трех глав.

Первая глава состоит из анализа предметной области, постановки задачи автоматизации процесса учета телефонных переговоров, а так же обзор существующих аналогов систем.

Вторая глава рассматривает процесс разработки программного продукта.

Третья глава содержит экономическое обоснование проектирования заявленной работы.

1 АНАЛИТИЧЕСКАЯ ЧАСТЬ

1.1 Анализ предметной области

1.1.1 Характеристика объекта автоматизации

В настоящей выпускной квалификационной работе рассматривается деятельность индивидуального предпринимателя ИП Земан П. С., основным видом деятельности которого является доставка готовых продуктов.

В соответствии с определением Налогового Кодекса РФ (п. 2, ст. 11), индивидуальный предприниматель — физическое лицо, зарегистрированное в установленном законом порядке и осуществляющее предпринимательскую деятельность без образования юридического лица.

К важной особенности осуществления предпринимательской деятельности в качестве индивидуального предпринимателя является тот факт, что гражданин отвечает по своим обязательствам всем принадлежащим ему имуществом, за исключением имущества, на которое в соответствии с законом не может быть обращено взыскание. В отличие, например, от участника общества с ограниченной ответственностью, где участник отвечает по обязательствам учреждённого им общества в основных случаях только в пределах своей доли в уставном капитале этого общества и ни в коем случае не своим личным имуществом.

Государственная регистрация и дальнейшая деятельность индивидуальных предпринимателей регламентируется Федеральным Законом РФ «О государственной регистрации юридических лиц и индивидуальных предпринимателей» N 129-ФЗ (ред. от 27.10.2020), Гражданским кодексом РФ, другими федеральными законами РФ, а также отдельными постановлениями Правительства РФ.

Целями деятельности предприятия ИП Земан П. С. являются расширение рынка товаров и услуг по продаже товаров, извлечение прибыли от своей деятельности.

Основные показатели деятельности компании приведены в таблице 1.1.

Таблица 1.1 – Показатели, характеризующие деятельность ИП Земан П. С.

Показатель	Единица измерения	2019 год	2020 год	Изменение в % в 2020 году относительно 2019 года
Объем реализации продукции	т.руб.	2100	3000	70%
Численность работающих	чел.	3	6	50%
Фонд заработной платы персонала	т.руб.	69	168	41%
Среднегодовая зарплата1-го работающего	т.руб.	23	28	82%

В ИП Земан П. С. заняты следующие сотрудники: директор (индивидуальный предприниматель), управляющий, поварской состав(3 штатные единицы), менеджер по приему заказов(2 штатные единицы), курьер(2 штатные едницы), обеспечивающий доставку заказов, программист.

Непосредственно ведением деятельности ИП, определением основных направлений работы, разработкой стратегии поведения на рынке занят сам индивидуальный предприниматель Земан П.С.

Рассмотрим далее обязанности каждого из сотрудников рассматриваемого ИП.

Главной задачей управляющего является ведение финансового учета имущества, обязательств и хозяйственных операций, своевременное и достоверное отражение результатов финансово-хозяйственной деятельности

компании. В своей работе управляющий руководствуется действующими законодательными актами РФ, инструкциями, приказами, распоряжениями, правилами, документами, утвержденными генеральным директором, регламентирующими работу управляющего.

В должностные обязанности управляющего входит:

– осуществление приема и контроля первичной документации по соответствующим участкам финансового учета и подготовка их к счетной обработке;

– расчет заработной платы;

– расчет налогов по зарплате по всем сотрудникам;

– подготовка отчета по фондам и отчет в налоговую инспекцию по единому социальному налогу;

– подготовка и сдача персонифицированного учета в пенсионный фонд, индивидуальных сведений по налогу на доходы физических лиц;

– оформление финансовых документов для их архивирования;

– выполнение работ по формированию, ведению и хранению финансовой информации.

К обязанностям менеджера по приему заказов относятся следующие:

– прием и обработка заказов клиентов, их оформление;

– осуществление информационной поддержки клиентов;

– осуществление информирования клиентов обо всех изменениях в ассортименте, увеличениях и снижениях цен, акциях по стимулированию спроса, времени прихода продукции на склад;

– окончательное согласование с клиентом условий по ценам, времени отправления и способу доставки продукции;

– участие в разработке и реализации проектов связанных с деятельностью отдела продаж;

– взаимодействие с подразделениями ресторана с целью выполнения возложенных задач;

– участие в рабочих совещаниях;

– ведение рабочей и отчетной документации;

– поддержание в активном состоянии данных о клиенте в информационной системе;

– участие в рассмотрении поступающих претензий и жалоб заказчиков;

– подготовка ответов на предъявленные иски, своевременное предъявление претензий потребителям при нарушении ими условий.

Повар выполняет следующие должностные обязанности:

– осуществляет прием на склад, взвешивание, хранение продуктов;

– руководит работой по погрузке, выгрузке продуктов их размещению;

– обеспечивает сохранность продуктов и соблюдение режимов хранения;

– участвует в проведении инвентаризации товарно-материальных ценностей;

– ведет учет складских операций;

– контролирует состояние техники безопасности и принимает меры к устранению выявленных недостатков, нарушений правил производственной санитарии, несоблюдения рабочими инструкций по охране труда.

Программист выполняет следующие должностные обязанности:

– устанавливает на серверы и рабочие станции сетевое программное обеспечение;

– конфигурирует систему на сервере;

– обеспечивает интегрирование программного обеспечения на файл-серверах, серверах систем управления базами данных и на рабочих станциях;

– поддерживает рабочее состояние программного обеспечения сервера и сайта;

– участвует в восстановлении работоспособности системы при сбоях и выходе из строя сетевого оборудования;

– готовит предложения по модернизации и приобретению сетевого оборудования;

– выполняет отдельные служебные поручения управляющего;

Основной обязанностью курьера является доставка заказов покупателям, а также доставка необходимых продуктов.

1.1.2 Постановка задачи автоматизации процесса учета телефонных переговоров

В настоящий момент в рассматриваемом ИП отсутствует возможность анализа телефонных звонков. Сотрудником отдела информационных технологий в конце периода по запросу начальника отдела или других ответственных лиц формируется список, где указывается общее количество и продолжительность разговоров по разделению на исходящие и входящие.

Такой список не является достаточно информативным, так как руководство учреждения нуждается в информации, которая бы позволяло определять, какие и кем звонки совершались в определенный момент времени, на какие номера, какой продолжительности, в какое время суток и так далее. Используемая в ИП автоматическая телефонная станция (Далее – АТС) не позволяет анализировать телефонные звонки.

Поэтому необходимо разработать такое программное обеспечение, которое учитывало все телефонные звонки и формировало соответствующие отчеты.

1.1.3 Обзор существующих аналогов и обоснование разработки информационной системы

В качестве аналогов разрабатываемой системы рассмотрим существующие системы автоматического учета телефонных переговоров: «Автоматизированная система учета телефонных подключений академии управления МВД России», ПО «ЭЛАР CRM», «PhoneBill», модуль автоматизации учета и обработки телефонных звонков в системе IP-телефонии, автоматизированная система учета услуг телефонной связи, Smart CollectCRM, модуль интеграции с АТС «Астериск 1.8».

«Автоматизированная система учета телефонных подключений академии управления МВД России»:

Программа предназначена для учета технических характеристик используемых каналов связи. Позволяет выбирать маршрут подключения с учетом различных технико-экономических критериев, в частности, длины маршрута, резервной кабельной емкости и т.д. Предоставляет возможность поиска оптимального маршрута между двумя точками сети, а также оптимизировать существующие линейные сети. Формы ввода технологического описания объектов дают возможности учитывать различные нестандартные способы подключений. Возможен импорт/экспорт данных в формат MS Excel. Программа позволяет автоматизировать задачи, связанные с учетом и идентификацией линий связи, повысить качество и производительность труда сотрудников отдела связи.

Тип реализующей ЭВМ: IBM PC-совмест. ПК.

Язык программирования: SQL, VB.NET.

Вид и версия операционной системы: Windows XP/Vista/7/10.

ПО «ЭЛАР CRM»:

Программа предназначена для автоматизации бизнес-процессов взаимодействия с клиентами и формирования «воронки» продаж. Программа обеспечивает: координацию действий различных отделов для взаимодействия с

клиентами; создание и ведение истории взаимоотношений клиента и организации; создание и ведение единого хранилища информации о взаимодействии с клиентами клиентской базы; использование многих каналов взаимодействия: телефонные звонки, электронная почта и т.д.; ведение актуальных контактов всех сотрудников, работающих по проекту или организации клиента, в том числе с использованием контактов Outlook; устранение неоднозначности при получении и отправке сообщений при привязке нескольких проектов или организаций клиента к одному контакту; автоматизированную централизованную обработку данных; оперативный доступ сотрудников организации к информации о взаимодействии с клиентами; учет и контроль проектов организации; управление организациями, проектами, контрактами и почтовыми сообщениями с возможностью формирования, поиска и просмотра информации; решение задач по ведению перечня мероприятий, резервированию и согласованию с руководителями работ по проекту; ведение справочников и классификаторов.

Тип реализующей ЭВМ: IBM PC - совмест. ПК.

Язык программирования: Visual C# .NET.

Вид и версия операционной системы: Windows Server от 2008 и выше, Юникс-подобные операционные системы: Centos 6 и выше, Ubuntu Server 14 и выше, Astra Linux Орел, возможна поддержка других Юникс-подобных операционных систем, которые поддерживают контейнер Java.

«PhoneBill»:

Программа является корпоративной программой, управляющей базой данных номеров телефонов и системой учета всей корпоративной телефонии. Содержит всю информацию (владелец, типы подключенных услуг, история владения, местонахождение настольного телефона, подключенные услуги, расходы, история звонков и т.д.) по номерам настольных телефонов, а также мобильной корпоративной связи. Детализация по звонкам, стоимости и т.д. вносится и обрабатывается ежемесячно из отчетов провайдеров с помощью встроенного модуля (импортер). Имеет три части: корпоративная телефонная книга,

ежемесячные отчеты по расходам, администраторская часть по управлению номерами и всей остальной информацией.

Тип реализующей ЭВМ: IBM PC-совмест. ПК.

Язык программирования: ASP.NET, VB.NET, Ruby.

Вид и версия операционной системы: Unix.

Модуль автоматизации учета и обработки телефонных звонков в системе IP-телефонии:

Программа для автоматизации учета и обработки телефонных звонков в системе IP- телефонии предназначена для решения коммуникационных задач компании. Область применения программы - обработка и учет телефонных звонков в системе IP-телефонии на предприятии. Программа для автоматизации учета и обработки телефонных звонков в системе IP-телефонии позволяет в значительной степени снизить трудозатраты на обработку и систематизацию звонков, увеличивая эффективность работы организации за счет автоматизации различных функций и многократного сокращения времени на их выполнение. Программа отражает способы решения коммуникационных задач компании и учитывает динамику обработки телефонных звонков организации, позволяя вести статистику звонков и составлять отчетность по ним в различных файловых форматах как эффективное средство оптимизации расходов и предоставление новых сервисов. Все данные хранятся в базе данных. Некоторые результаты работы программы, как части системы, представлены с помощью скриншотов.

Тип реализующей ЭВМ: IBM PC-совмест. ПК.

Язык программирования: PHP, HTML5, Java, SQL.

Вид и версия операционной системы: Windows XP/Vista/7/10, AsteriskNOW или любая другая операционная система GNU/Linux, поддерживающая работу платформы Asterisk.

Автоматизированная система учета услуг телефонной связи:

Программа предназначена для: поддержки процессов обслуживания абонентов сети телефонной связи; учета объема и номенклатуры предоставленных

услуг телефонной связи и расчета их стоимости по типам оказываемых услуг (услуги местной, междугородной, международной телефонной связи); расчета затрат с целью контроля использования и оптимизации расходов на услуги телефонной связи образовательного учреждения. Область применения программы: абоненты учрежденческой телефонной сети; телекоммуникационные подразделения, обеспечивающие доступ к услугам электросвязи; финансовые подразделения. Функциональные возможности: сбор первичных данных с учрежденческих автоматических телефонных станций; обработка, агрегация, расчет стоимости услуг связи; привязка к абонентам; сохранение данных за отчетный период; справочно-информационное обслуживание абонентов; подготовка отчетных документов; взаимодействие с внешними автоматизированными системами по импорту установочных данных и экспорту результатов расчетов.

Тип реализующей ЭВМ: IBM PC-совмест. ПК.

Язык программирования: Visual Basic 6.0 (SP5).

Вид и версия операционной системы: Windows XP/Vista/7/10.

Smart CollectCRM:

Программа предназначена для сопровождения процесса взыскания просроченной задолженности физических, юридических лиц в банковской сфере, сфере работы коллекторских агентств и микрофинансовых организаций. Функциональные возможности: работа со стратегиями Soft Collection (телефонное взыскание), Hard Collection (выездное взыскание), судебное взыскание, исполнительное производство; управляемый список клиентов и задач (с гибкими настойками фильтрации и сортировки) с возможностью добавления полей для администрирования задач; автоматическая подгрузка данных из реестров клиентов с учетом добавления новых реестров и внесения изменений в существующие реестры; автоматическая подгрузка информации о платежах; возможность сохранения и хранения копий документов для клиентов в ПО; формирование выходных форм (писем, реестров) по шаблонам; формирование стандартных СМС сообщений по шаблонам.

Тип реализующей ЭВМ: IBM PC-совмест. ПК, Macintosh.

Язык программирования: PHP, JavaScript.

Вид и версия операционной системы: Windows, Linux, MacOS.

Модуль интеграции с АТС «Астериск 1.8»:

Программа предназначена для автоматизации работы по учету входящих и исходящих телефонных звонков. Программа может применяться в коммерческих и технических службах предприятий телекоммуникационной отрасли. Программа обеспечивает выполнение следующих функций: при поступлении входящего звонка: автоматическое создание электронного документа для ввода информации о контакте с указанием телефонного номера входящего звонка; поиск контакта в базе данных по телефонному номеру и автоматическое заполнение реквизитов контактного лица в созданном документе, если удалось найти соответствующий телефонный номер в базе данных; при выполнении исходящего звонка: автоматический набор номера из базы данных контактов и из электронного документа для ввода информации о контакте; формирование отчетности о входящих и исходящих звонках.

Тип реализующей ЭВМ: IBM PC - совмест.ПК.

Язык программирования: Встроенный язык программирования 1С: Предприятие.

Вид и версия операционной системы: Windows, Linux, MacOS.

В связи со сложностью используемых технологий, однозначно понадобится настройка комплекса продавцом, а также обучение пользователей. Часто, сумма, которую необходимо заплатить на настройку программы, уже первоначально превышает стоимость самой программы, что ставит под сомнение необходимость ее приобретения.

Кроме того, при покупке такой программы фирма получает излишнюю функциональность, которая выражается в наличии множества ненужных функций, которые, возможно, никогда и не будут востребованы.

Следовательно, покупка такой системы не будет оправдана ни с какой точки зрения в рамках решения поставленной задачи автоматизации.

Поэтому наиболее логичным способом решения существующей проблемы будет в данном случае самостоятельная разработка системы автоматизации учреждения, которая будет учитывать все особенности автоматизируемых бизнес-процессов и наиболее полно отвечать существующим потребностям.

1.2 Разработка требований к информационной системе

1.2.1 Функциональные требования

С помощью разрабатываемого программного обеспечения необходимо получать информацию о том, в какое время, по какому номеру звонил любой из сотрудников и сколько длился разговор. Приложение должно позволять составлять и печатать отчеты и графические диаграммы.

Программа должна позволять:

Производить тарификацию переговоров:

– по номерам внутренних телефонов;

– по внешним (городским) линиям мини-АТС;

– по наиболее часто набираемым номерам.

Формировать следующие отчеты:

– загруженность внешних и внутренних линий (детально и в виде графических диаграмм);

– принятые и неотвеченные вызовы;

– оперативность приема звонков;

– распределение входящих звонков по абонентам и часам;

– списки городских и внутренних номеров.

В отчет должна выводится информация о произведенных входящих и исходящих звонках (город, с какой внешней линии был произведен звонок,

набранный номер, внутренний номер, стоимость, дата и время произведенного звонка, продолжительность разговора).

1.2.2 Нефункциональные требования

1.2.2.1 Требования к аппаратному и программному обеспечению

Для нормального функционирования информационной системы сервер баз данных должен удовлетворять техническим требованиям, предъявляемым к аппаратному обеспечению, а именно:

– процессор не ниже Intel Core i3 не ниже 3.30 GHz. Рекомендуется использовать многопроцессорные сервера;

– оперативная память не менее 4096 Мбайт (рекомендуется 8128 Мбайт и выше). Компьютер конечного пользователя должен удовлетворять следующим техническим требованиям:

– Intel Celeron Dual-Core, 2000Mhz, 1024 Мб RAM, 160 Гб HDD, VGA видеоадаптер, монитор LCD, клавиатура, мышь.

Для обеспечения сетевой работы информационной системы компьютеры возможных пользователей системы должны быть объединены в локальную вычислительную сеть. В качестве сетевого интерфейса необходимо применить сетевой протокол TCP/IP с пропускной способностью 10/100 Мбит/с.

Программный продукт должен позволять одновременное использование несколькими пользователями(не более чем 10) одновременно, по количеству информационной системы.

Для работы с системой компьютер конечного пользователя должен быть оснащен операционной системы не ниже Microsoft Windows 7 Professional.

1.2.2.2 Количественные требования

– максимальное время выполнения функций по вводу в систему справочных данных и данных для документов не должно превышать пяти минут для одной операции;

– редактирование данных справочников и форм документов не должно превышать пяти минут для одного документа;

– функция вывода данных (формирования отчетов) не должна превышать пяти минут для одного отчета;

– количество записей в базе данных – около 150000.

1.2.2.3 Требования к защите информации

Для обеспечения информационной системы от несанкционированного доступа необходимо предусмотреть авторизацию пользователя при входе в систему.

Администратор при резервном копировании и архивировании информации базы данных должен защитить ее служебным паролем.

В системе требуется предусмотреть возможность разграничения полномочий пользователей. Необходимо, чтобы администратор мог распределять уровни доступа к информации каждого из пользователей и наделять им права доступа к системе.

В системе должны быть предусмотрены две группы пользователей:

– администратор, обладающий всеми правами по изменению данными в программе и получению всех отчетов;

– пользователь, выполняющий функции в рамках своего аккаунта.

1.2.2.4 Требования к надежности

Должна обеспечиваться сохранность информации при наступлении следующих событий:

– отказ оборудования рабочей станции, в случае хранение данных на серверах АИС;

– отключение питания на сервере баз данных;

– отказ линий связи;

– отказ аппаратуры сервера (процессор, накопители на жестких дисках).

Средствами обеспечения сохранности информации при авариях и сбоях в процессе эксплуатации являются:

– носители информации (сменные: оптические-дисковые или магнитные - ленточные, накопители на сменных жестких дисках);

– создание резервной копии базы данных;

– создание резервной копии программного обеспечения.

Для восстановления данных и программного обеспечения из резервной копии должны использоваться средства резервного копирования и архивирования.

АИС должна обеспечивать возможность резервирования всех данных, хранящихся на серверах АИС, а также возможность их восстановления.

Резервное копирование данных должно осуществляться администратором АИС ежедневно, автоматически по расписанию.

Должна быть предусмотрена возможность восстановления данных за день сбоя с помощью их повторного ввода или импорта (для данных из внешних систем, получаемых автоматически).

Под критерием отказа понимается признак или совокупность признаков, установленных в нормативно-технической и (или) проектно-конструкторской документации и позволяющих определить наличие отказа в некоторой функции автоматизированной системы. Признаком отказа в выполнении некоторой функции системы является нарушение признаков нормальной работы (своевременности и достоверности решения задач) и т.д.

Перечень возможных отказов и причины этих отказов для функций автоматизированной системы определяется на этапе технического проектирования.

Коэффициент готовности – не менее 0,9.

1.2.2.5 Эксплуатационные требования

Система должна обеспечивать непрерывный круглосуточный режим эксплуатации с учетом времени на техническое обслуживание.

В помещениях, предназначенных для эксплуатации системы, должны отсутствовать агрессивные среды, массовая концентрация пыли в воздухе должна быть не более 0,75 мг/м^3, электрическая составляющая электромагнитного поля помех не должна превышать 0,3 в/м в диапазоне частот от 0,15 до 300,00 МГц.

Напряжение питания сети должно быть 220 ±10.

Климатические факторы помещения для эксплуатации системы должны быть по ГОСТ 15150-69.

Нормальными климатическими условиями эксплуатации системы являются:

– температура окружающего воздуха (20±5)°С;

– относительная влажность окружающего воздуха (60±5)°С при атмосфере воздуха (20±5)°С;

– атмосферное давление (101,3±4) Кпа (760±30) мм.рт.ст.

Система должна сохранять работоспособность при воздействии следующих климатических факторов:

– температура окружающего воздуха от 10°С до 35°С;

– относительная влажность воздуха от 40 до 80% при температуре 25°С.

1.2.2.6 Требования к эргономике

Система обязана иметь русскоязычный интерфейс на всех стадиях набора, анализа, получения и передачи информации, который поможет пользователю свободно ориентироваться в рабочем пространстве.

Система должна быть создана с учетом современных требований по эстетике и эргономике. Обычно такие требования включают в себя:

– удобство и комфорт работы пользователя;

– адекватный интерфейс, реализованный с учетом типичных для пользователя задач;

– расположение пунктов меню или их аналогов в рамках указанных функций, задач и технологий работы пользователей;

– полное понимание назначения пунктов меню или его аналога;

– применение графического пользовательского интерфейса;

– вся цветовая схема должна быть выдержана в спокойных тонах, не вызывающих переутомление зрения;

– полное соответствие интерфейса программного обеспечения стандартам, принятым в операционной системе MS Windows;

– указание критериев для выполнения поиска и выборки данных без привлечения языков программирования;

– доступность полноценного набора используемых словарей и справочников;

– реализация аналогичных функций схожими методами.

Интерфейс должен разрабатываться в соответствии с требованиями юзабилити.

Юзабилити является свойством системы или продукта. Если продукт можно назвать юзабельным, то значит, что пользователи достигают цели в процессе применения этого продукта; это значит, что нет границ в решении задач. Специалисты данной сферы должны учитывать в своей работе эти задачи и цели.

Выводы по главе один:

Важным фактором при разработке любого программного обеспечения является точная формулировка цели проекта и анализ предметной области.

В этой главе обсуждаются и анализируются подходы к разработке информационной системы, проблемы, возникающие при создании отчета телефонных звонков, инструменты разработки и необходимость автоматизации учета телефонных разговоров.

Для решения этих проблем было решено разработать информационную систему. Знание предметной области и анализ, выполненный в этой главе, являются предпосылкой для построения системы.

2 РАЗРАБОТКА ИНФОРМАЦИОННОЙ СИСТЕМЫ УЧЕТА РАЗГОВОРОВ СОТРУДНИКОВ

2.1 Моделирование информационной системы

2.1.1 Функциональные модели IDEF0

После разработки внедрения информационной системы по учету телефонных разговоров сотрудников технология основных процессов, которая описана выше, изменится. Рассмотрим, как будет осуществляться учет телефонных разговоров сотрудников в проектном варианте.

На Рисунке 2.1 представлена схема, характеризующая процесс учета телефонных разговоров сотрудников «To-be».

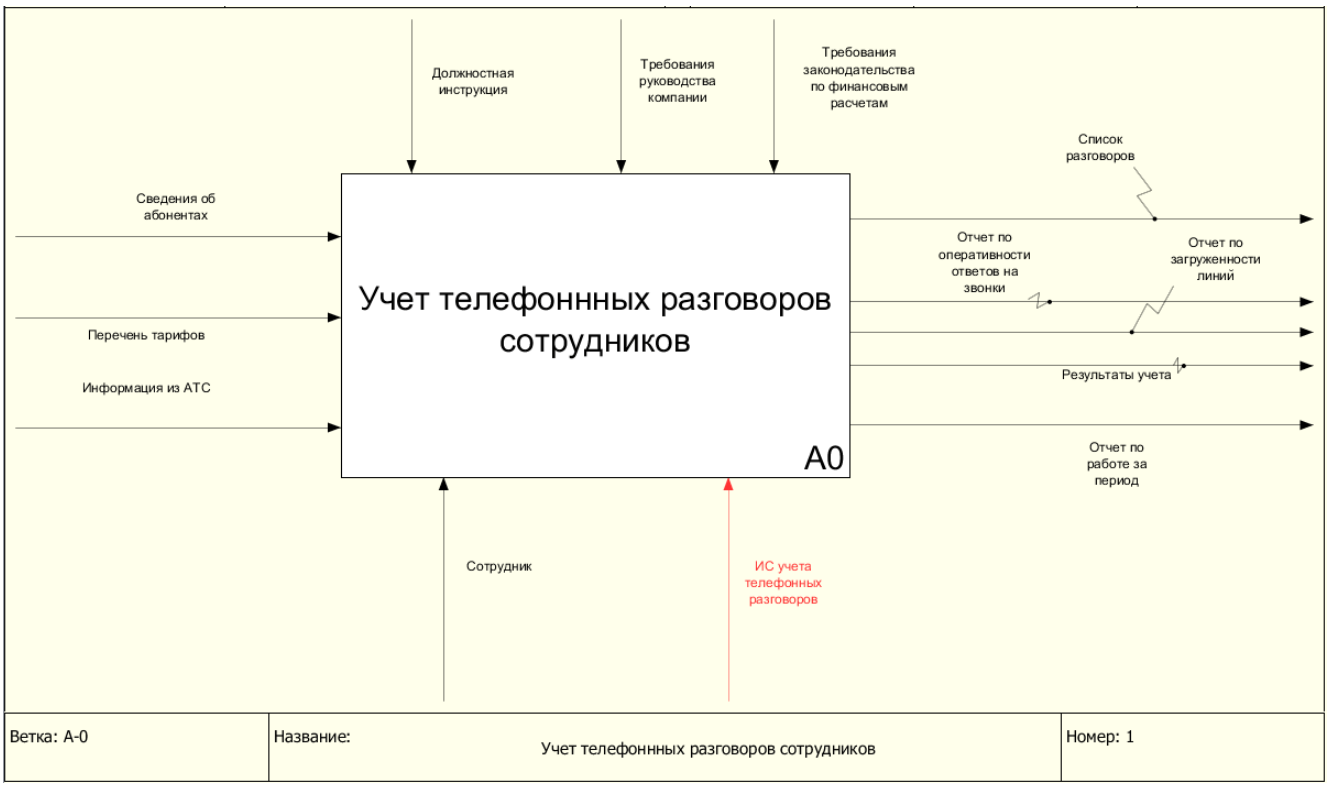

Рисунок 2.1 – Учет договоров и расчетов с клиентами «To-be»

Таким образом, в данном случае (по сравнению с функциональной моделью «As-is»), добавляется новый механизм — информационная система учета телефонных разговоров. В результате внедрения данной информационной системы пропадает необходимость использования ручного труда сотрудника, так как все данные в систему попадают в результате работы с программы с АТС, пользователь же получает отчетные документы с необходимой периодичностью.

На Рисунке 2.2 представлена схема декомпозиции процесса телефонных разговоров сотрудников.

Данный процесс включает в себя также три процесса второго уровня:

– учет абонентов, тарифов;

– настройка программы для интеграции с АТС;

– непосредственный учет телефонных разговоров сотрудников;

– получение отчетных документов.

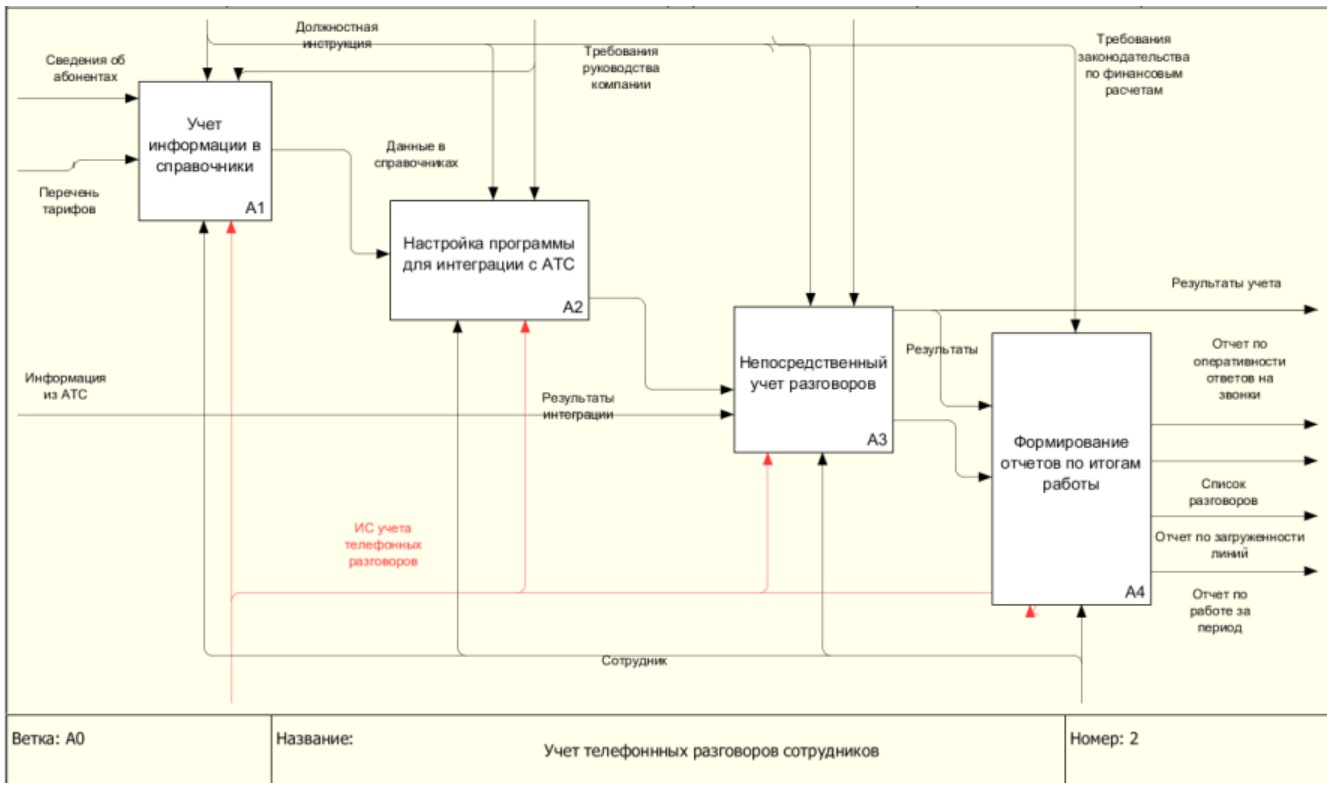

Рисунок 2.1 – Схема декомпозиции процесса телефонных разговоров сотрудников «To-be»

Сами процессы остались неизменными (по сравнению со схемой КАК ЕСТЬ), но их внутреннее содержание изменилось.

Получение отчетов заключается в переходе на экранную форму с соответствующей формой сортировки, инициализации подготовки отчета и получении его в табличном и графическом виде.

2.1.2 Объектное моделирование

2.1.2.1 Диаграмма вариантов использования

В процессе разработки системы ее функциональные возможности – поведение, которое ей присуще с точки зрения заказчика, записывается при помощи модели прецедентов. Эта модель отражает, какие функции может реализовать система, а также где она может работать. Модель прецедентов

создается на ранних этапах разработки и помогает заказчику четко сформировать свои требования, а инженеру-разработчику – суметь понять, что же хочет заказчик.

Базовыми элементами модели прецедентов становятся прецеденты и актеры.

Субъекты можно внедрять в информационную систему, получать данные из системы или делать сразу вместе 2 операции. В роли субъекта выступает и другая система, если она сможет работать с создаваемой системой, и не станет ее неотъемлемой частью.

Прецеденты необходимы для моделирования диалога между субъектами и самой системой. Они дают те возможности, которые система обеспечивает отдельному субъекту.

В схеме прецедентов 1 актер – пользователь информационной системы учета телефонных разговоров сотрудников.

На Рисунке 2.3 представлена диаграмма прецедентов, которая представляет функции администратора по работе с системой.

Рисунок 2.3 – Диаграмма прецедентов системы

К основным функциям пользователя относятся:

– авторизация пользователя для входа в программу;

– работа со справочниками, то есть ввод в систему данных о тарифах, абонентах, типах линии, статусах соединения;

– интеграция программы с АТС;

– работа с отчетами.

В виде отчетов администратор может получить следующие документы:

– отчет по всем разговорам;

– отчет по загруженности линии;

– отчет по принятым и не отвеченным вызовам;

– отчет по оперативности приема звонков.

2.1.2.2 Диаграмма классов

Диаграмма классов (англ. Static Structure diagram) — структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их кооперaций, атрибутов (полей), методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования[1].

Целью создания диаграммы классов является графическое представление статической структуры декларативных элементов системы (классов, типов и т. п.) Она содержит в себе также некоторые элементы поведения (например — операции), однако их динамика должна быть отражена на диаграммах других видов (диаграммах коммуникации, диаграммах состояний). Для удобства восприятия диаграмму классов можно также дополнить представлением пакетов, включая вложенные[2].

При представлении сущностей реального мира разработчику требуется отразить их текущее состояние, их поведение и их взаимные отношения. На каждом этапе осуществляется абстрагирование от маловажных деталей и концепций, которые не относятся к реальности (производительность, инкапсуляция, видимость и т. п.). Классы можно рассматривать с позиции различных уровней. Как правило, их выделяют три основных: аналитический уровень, уровень проектирования и уровень реализации[3]:

– на уровне анализа класс содержит в себе только набросок общих контуров системы и работает как логическая концепция предметной области или программного продукта;

– на уровне проектирования класс отражает основные проектные решения касательно распределения информации и планируемой функциональности, объединяя в себе сведения о состоянии и операциях;

– на уровне реализации класс дорабатывается до такого вида, в каком он максимально удобен для воплощения в выбранной среде разработки; при этом не воспрещается опустить в нём те общие свойства, которые не применяются на выбранном языке программирования.

Диаграмма классов приложения приведена на рисунке 2.4.

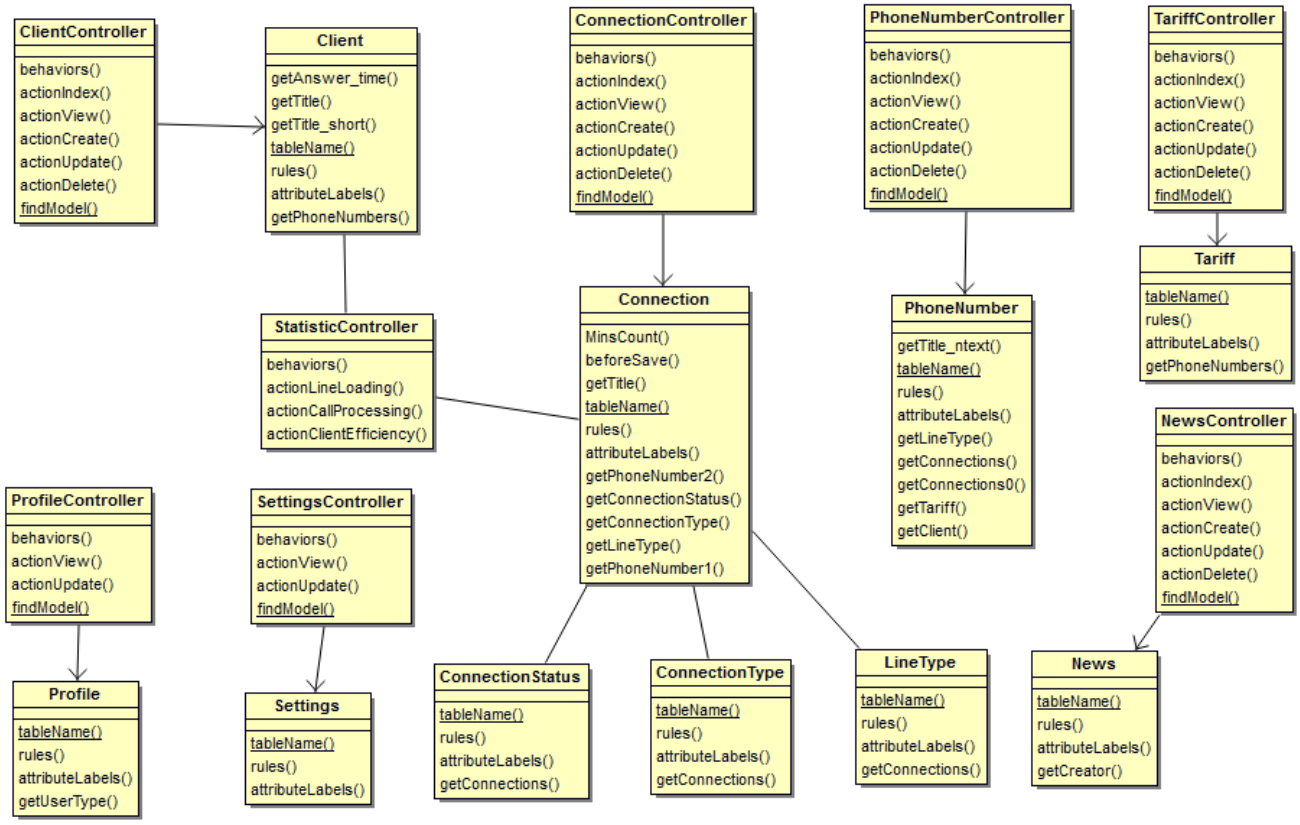

Рисунок 2.1 – Диаграмма классов

ForecastSettingsController реализует методы редактирования (CRUD, сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») модели «ForecastSettings».

LinkController реализует методы редактирования (CRUD, сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») для модели «Link».

«UserController» реализует методы редактирования (CRUD, сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») модели «User».

2.1.2.3 Диаграмма последовательности

Прецедент «Аутентификация пользователя» активизируется всеми субъектами информационной системы. Прежде чем пользователю начать работу с базой данных, система запрашивает его пароль и логин. Если пользователь не зарегистрирован или в пароле и/или логине допустил ошибку, то он не получает доступа к работе в информационной системе [6]. После успешной проверки логина и пароля пользователя открывается главное окно программы. На Рисунке 2.5 представлена диаграмма последовательности, и на Рисунке 2.6 диаграмма действий этого прецедента.

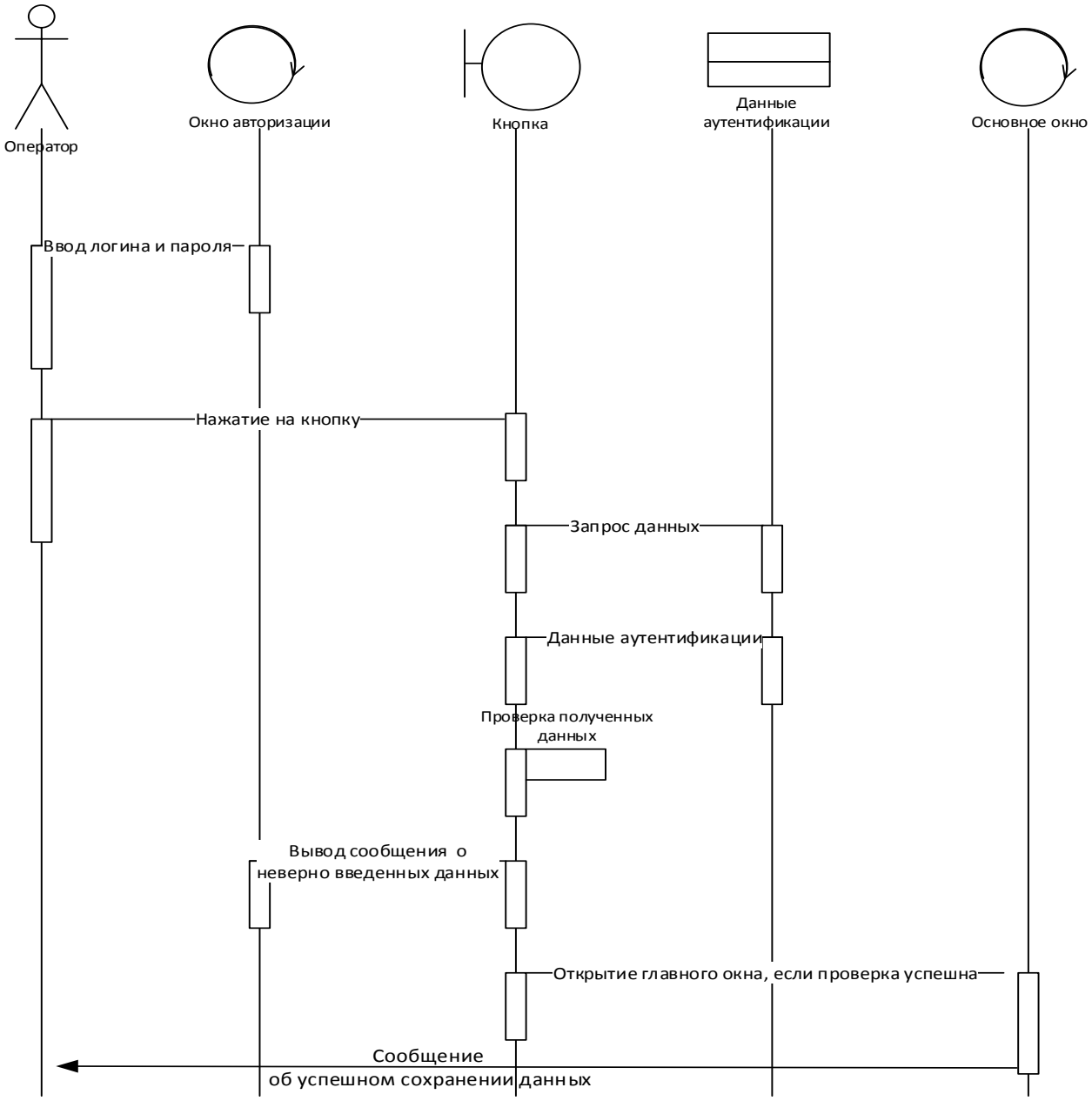

Рисунок 2.2 – Диаграмма последовательности «Аутентификация пользователя»

Прецедент «Учет тарифа» активизируется субъектом информационной системы пользователь. Данный прецедент описывает процесс ввода новых данных в базу данных. Данные вводятся в формы ввода на основании документов, а также вся справочная информация, которая нужна в процессе работы с системой.

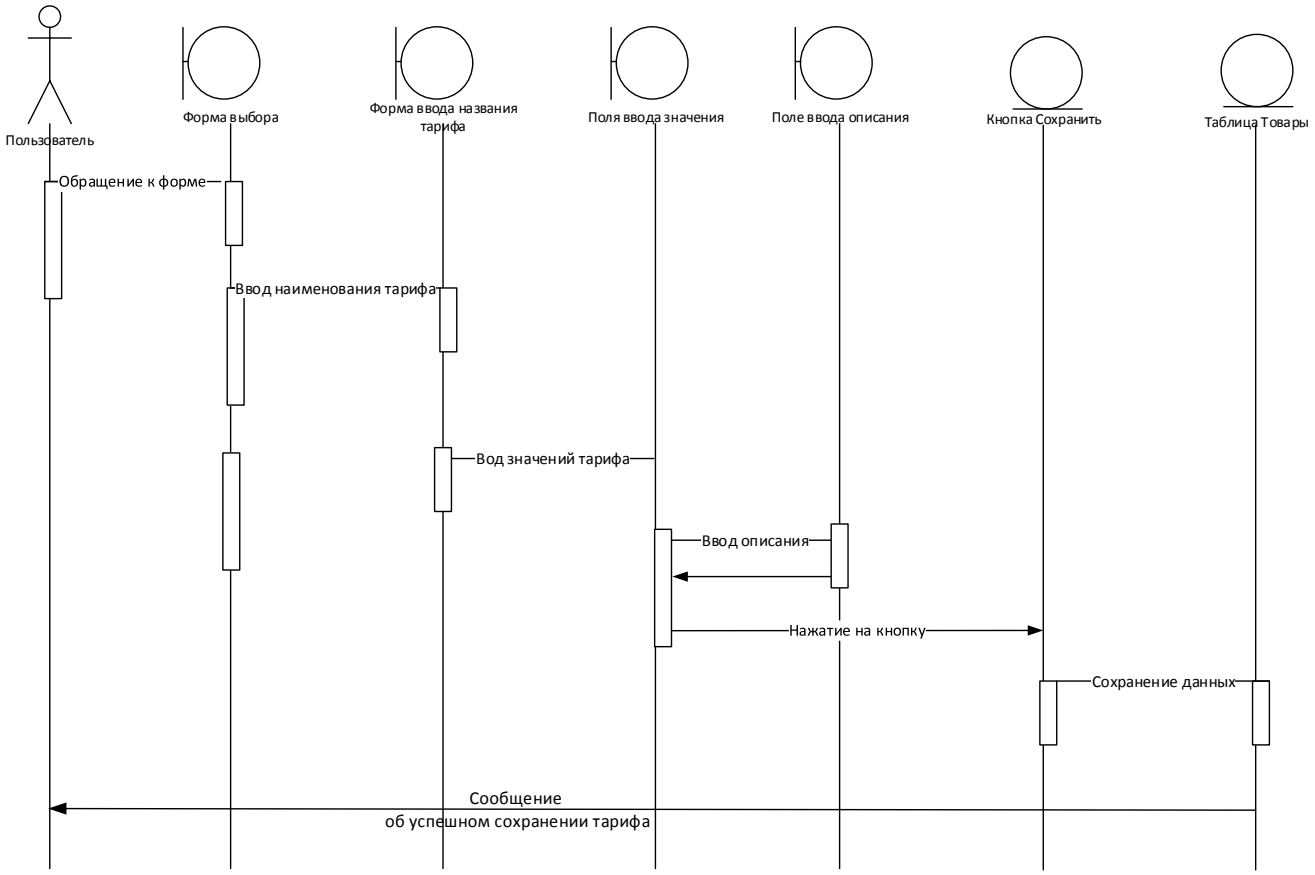

Рисунок 2.6 Диаграмма последовательности «Учет тарифа»

Учет тарифа происходит путем учета наименования тарифа, значений тарифа, сохранение в базе данных.

2.1.2.4 Диаграмма состояний

Диаграммы состояний применяют для описания поведения, которое возможно в рамках варианта применения или поведения объектов, компонента, узла или самой системы. Поведение описывается через автомат, отражающий доступные периодичности состояний экземпляра сущности и возможные переходы между ними в рамках его жизненного цикла, начиная от реализации и заканчивая удалением.

Диаграмма состояний (автомат) является связанным ориентированным графом, вершинами которого выступают состояния, а дуги необходимы для

визуализации переходов из одного состояния в другое. Само состояние подразумевает ситуацию в ходе жизни экземпляра сущности, когда такая ситуация полностью соответствует отдельному условию, экземпляр реализует отдельные операции или ждет реализации какого-то события. К примеру, для объекта его указывается в рамках совокупности отдельных значений атрибутов, при этом корректировка этих значений отражает и изменение состояния моделируемого объекта.

Диаграмма состояний приведена на Рисунке 2.7.

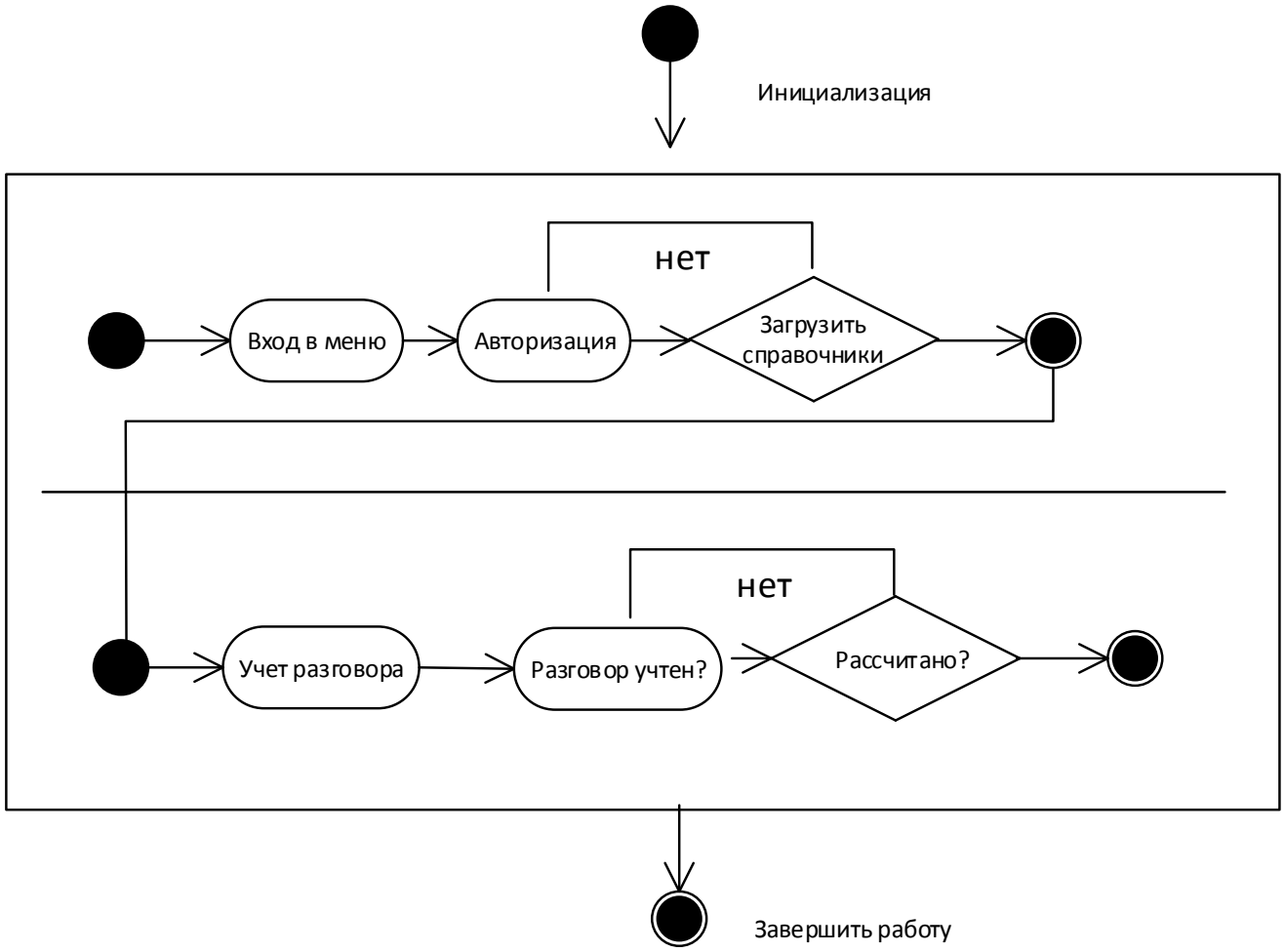

Рисунок 2.7 – Диаграмма состояний авторизации пользователя

2.1.2.5 Диаграмма деятельности

Диаграмма деятельности выступает в роли UML диаграммы, отражающей разложение отдельного вида деятельности на её несколько частей. Сама деятельность представляет из себя спецификацию реализуемого поведения в виде упорядоченного последовательного и параллельного выполнения элементов подчинения — внутренних видов деятельности и конкретных действий, которые соединены между собой потоками, идущими от выходов одного узла ко входам другого.

Диаграммы деятельности необходима при моделировании технологических и бизнес процессов, параллельных и последовательных вычислениях.

Диаграмма деятельности получения списков приведена на Рисунке 2.8.

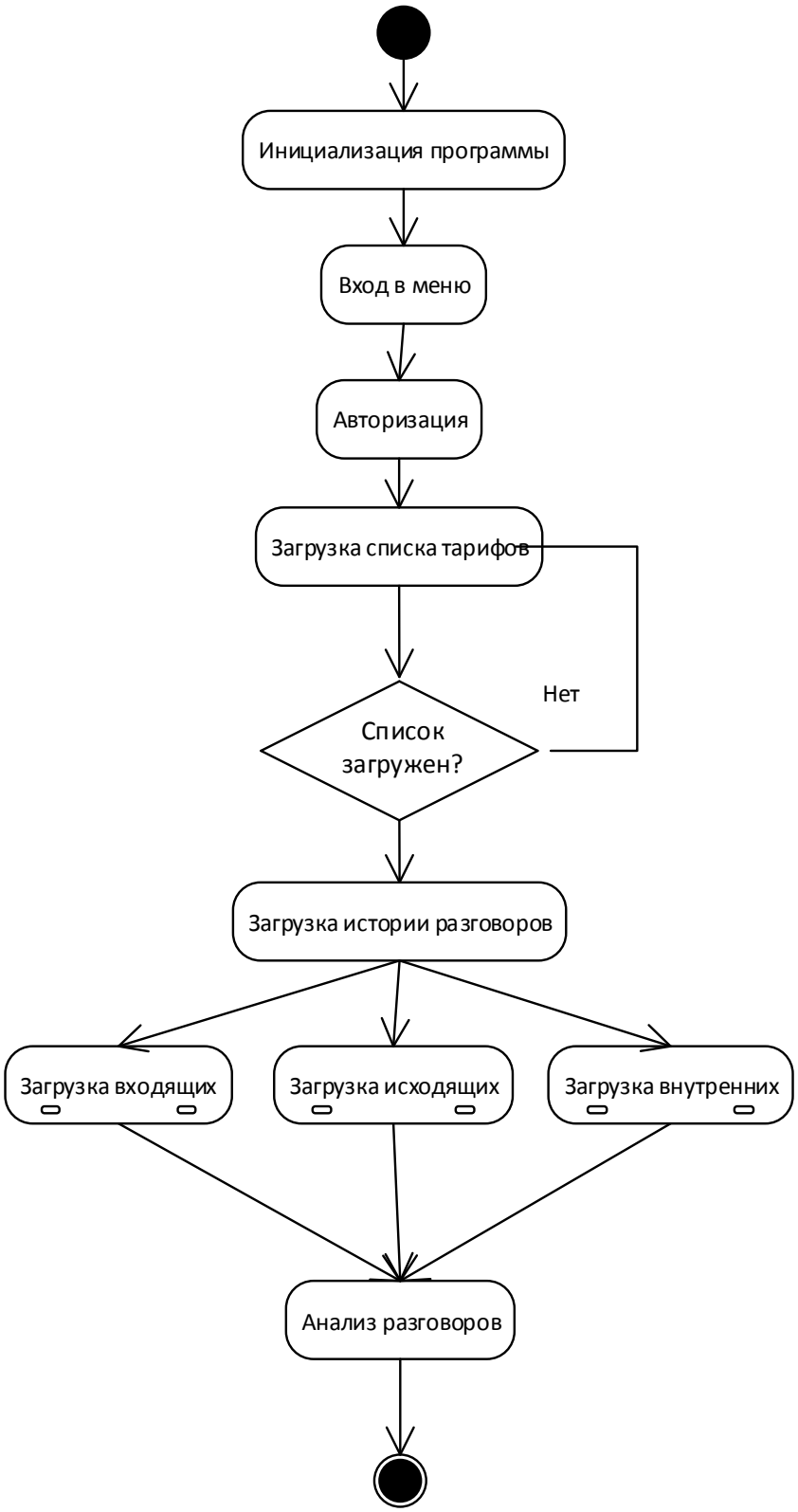

Рисунок 2.8 – Диаграмма деятельности получения отчетов

2.1.2.6 Диаграмма компонентов

Диаграмма пакетов показывает, из каких частей состоит проектируемая система и как эти части связаны друг с другом [32].

Пакет – совокупность описаний классов и других программных ресурсов, в том числе и самих пакетов.

Анализ концептуальной модели позволил выделить следующие пакеты:

– интерфейсные элементы – классы, реализующие интерфейсные компоненты;

– пользовательский интерфейс – классы, реализующие объекты интерфейса с пользователем;

– интерфейс с базой данных – классы, реализующие интерфейс с базой данных;

– база данных.

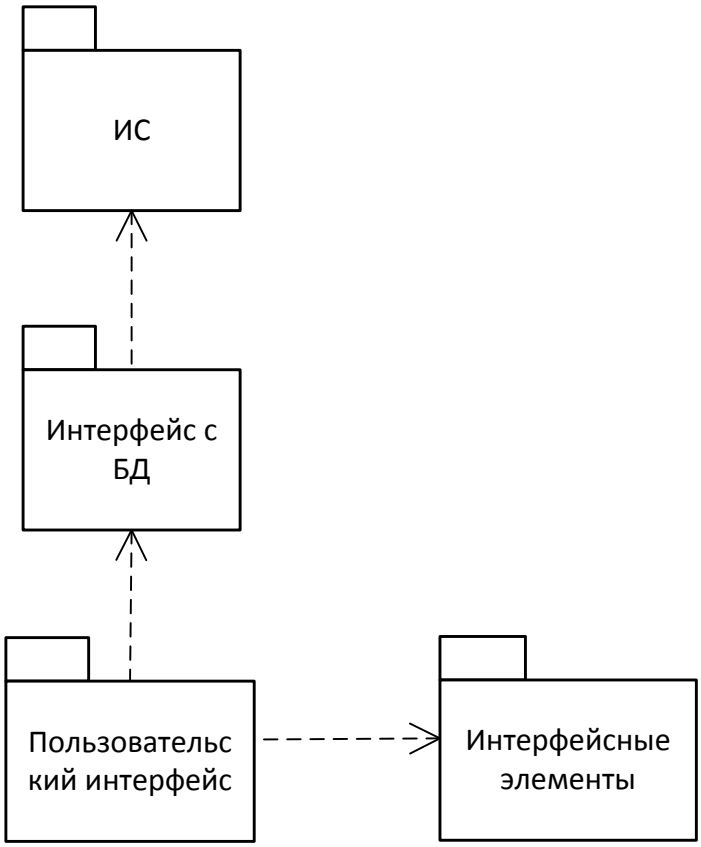

Рисунок 2.3 – Диаграмма пакетов

2.1.2.7 Диаграмма развертывания

Диаграмма развертывания системы представлена на рисунке 2.10.

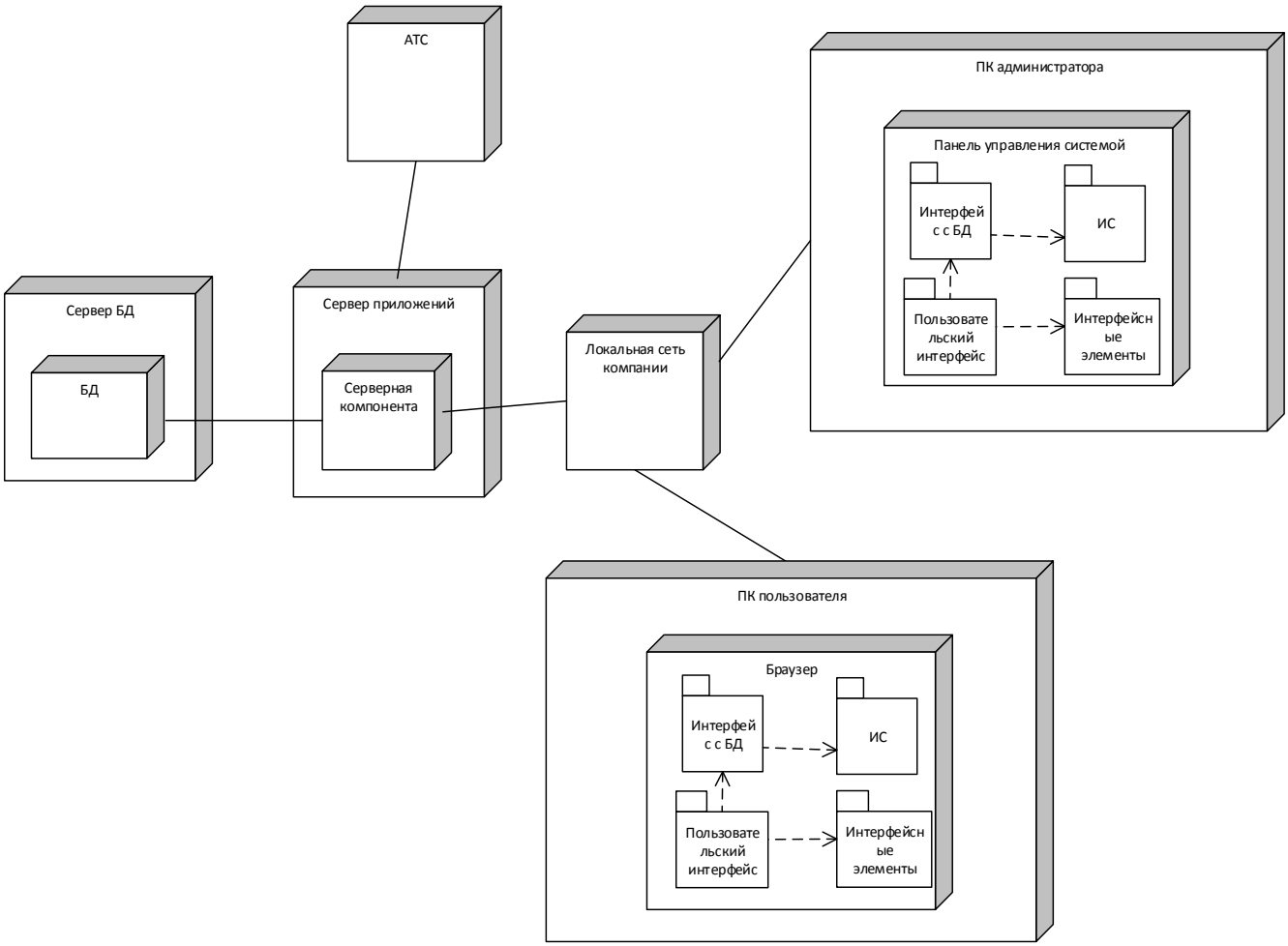

Рисунок 2.4 – Диаграмма развертывания

Как следует из приведенной диаграммы, для развертывания системы необходим сервер баз данных, сервер приложений, АТС, а также персональные компьютеры пользователей системы с установленными на них браузерами.

2.2 Разработка модели данных информационной системы

2.2.1 Логическая модель данных

Есть два основных подхода к разработке систем баз данных: снизу-вверх или сверху вниз. Первый подходит для создания небольших баз данных с ограниченным числом атрибутов. Использование такого подхода усложняется при создании баз данных с огромным числом атрибутов, описать среди которых все доступные функциональные зависимости проблематично. При создании сложных систем баз данных лучше всего применять нисходящий подход, хорошо зарекомендовавший себя в рамках модели «сущность-связь». Тут работа связана с определением сущностей и процессом их взаимодействия, которые очень важны для подобной разработки.

Весь путь создания базы данных включает 3 фазы: концепция, логическая модель и проектирование прототипа. Любая фаза состоит из необходимой модели данных, которая становится источником данных для другой фазы. Основное значение тут возложено на концепцию, реализуемую в ракмкх параметров, указанных в спецификации пользовательских требований. Подготовка концепции баз данных никак не пересекается в такие подробности ее реализации, как тип применяемой целевой СУБД, тип используемой вычислительной платформы и т.п., но качество концепции тут уже имеет решающее значение, позволяющее отразить трудозатраты на создание системы, ее скорость работы и текущий успех. Опыт создания и использования информационной системы говорит о том, что ошибки, возможные в процессе этого этапа, очень трудно выявить и устранить, поскольку они встречаются обычно уже на последующих этапах создания системы – при реализации и поддержке.

На этапе проектирования логической модели концепция данных переходит в логическую модель, создаваемую в рамках указанной модели хранения информации основной СУБД. По итогу, этот этап отражает, какая СУБД используется в качестве целевой – иерархическая, сетевая, реляционная или объектно-ориентированная. Тут проходят все остальные аспекты начальной СУБД – к примеру, некие особенности физической реализации хранения данных. Логическая модель, отражающая особенности отображения о создаваемой системе больше одного типа пользователей, считается глобальной логической моделью данных. Есть пара базовых подходов для создания совокупной логической модели: метод внедрения представлений и централизованный способ. Если создается крупная информационная система, лучше и эффективнее использовать второй подход, когда общая логическая модель реализуется методом слияния отдельных моделей, показывающих представления отдельных групп пользователей.

В процессе создания прототипа принимаются решения о вариантах реализации текущей базы данных. Поэтому физическое проектирование очень сильно завязано на конкретной СУБД. Между разработкой логической модели и подготовкой прототипа есть постоянная обратная связь, т.к. все решения, принимаемые на этапе создания прототипа для повышения производительности системы, очень влияют и на структуру логической модели. Цель создания прототипа базы данных заключается в описании варианта итоговой реализации проекта всей базы данных.

Для минимизации чрезмерности технологических процедур в рамках выполнения системных функций, все совокупные для нее процедуры быть создаваться по одному прототипу.

Все решения системы по проекту при реализации разного рода задач системы должны поддерживать:

– следование единым правилам реализации взаимодействия с пользователем;

– одинаковую реакцию системы на ошибочные действия пользователя;

– одинаковый вид заполнения классификаторов с применением справочников;

– применение единого перечня терминов и определений Системы при построении диалога и генерации экранов;

– идентичный подход в рамках разделения доступа пользователей к данным системы.

Инфологическая (концептуальная) модель подразумевает описание предметной области, реализованное безотносительно к применяемым программным и техническим средствам. Инфологическая модель должна поддерживать простую корректировку и быть динамической. К базовым требованиям, которые могут предъявляться к инфологической модели, относят следующие:

– модель должна включать в себя всю необходимую и достаточную информацию для беспроблемного проектирования базы данных;

– модель должна быть проста и понятна для тех лиц, которые участвуют в создании системы.

ER-модель выражается как логическая структура данных об объектах системы. Компонентами ER-модели становятся сущности (объекты) и отношения (связи объектов между собой). Объект содержит множество реализаций или экземпляров. Экземпляр объекта создается при помощи совокупности конкретных значений реквизитов и должен однозначно определяться, т.е. выражаться значением ключа объекта, состоящего из нескольких (одного либо более) ключевых реквизитов.

Сущности могут выступать как независимыми, так и зависимыми. Независимыми именуют такие сущности, где любой выбранный экземпляр однозначно определяется в структуре его отношений с другими сущностями. Однозначное определение экземпляра зависимой сущности зависит от взаимосвязи с другими сущностями.

Для отражения таких отношений между сущностями применяются связи. Связь возможна, если экземпляры сущностей взаимосвязаны логически.

В результате анализа предметной области были выделены сущности, которые необходимы для функционирования программы.

К числу таких сущностей относятся:

– Абонент.

– Соединения.

– Тип линии.

– Статус соединения.

– Телефонный номер.

– Пользователь.

– Тип пользователя.

– Тариф.

– Сообщения.

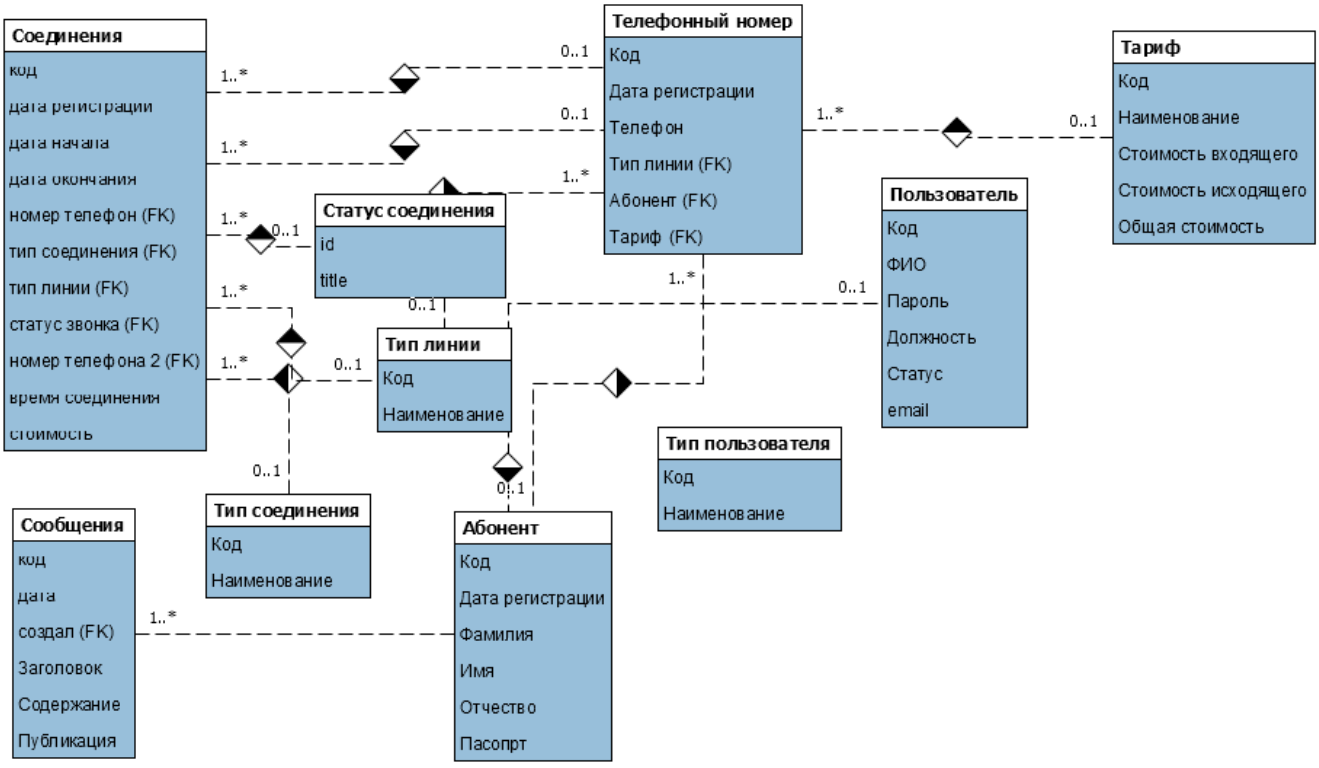

Рисунок 2.1 – Логическая модель базы данных

Все отношения в базе данных находятся в 3 нормальной форме, а, следовательно, и в 1 и во 2, т.к.:

– само отношение существует в 3 форме: каждый не начальный атрибут транзитивно связан с начальным ключом;

– само отношение существует в 1 форме, поскольку отношение нормализовано, то есть любой простой атрибут имеет атомарные неделимые значения, присутствует ссылочная полноценность, другими сломами, внешнему ключу ставится в соответствие строка объектного отношения, в противном случае внешний ключ обрушался бы к неизвестному объекту.

2.2.2 Физическая модель данных

Физическая модель базы данных приведена на Рисунке 2.12.

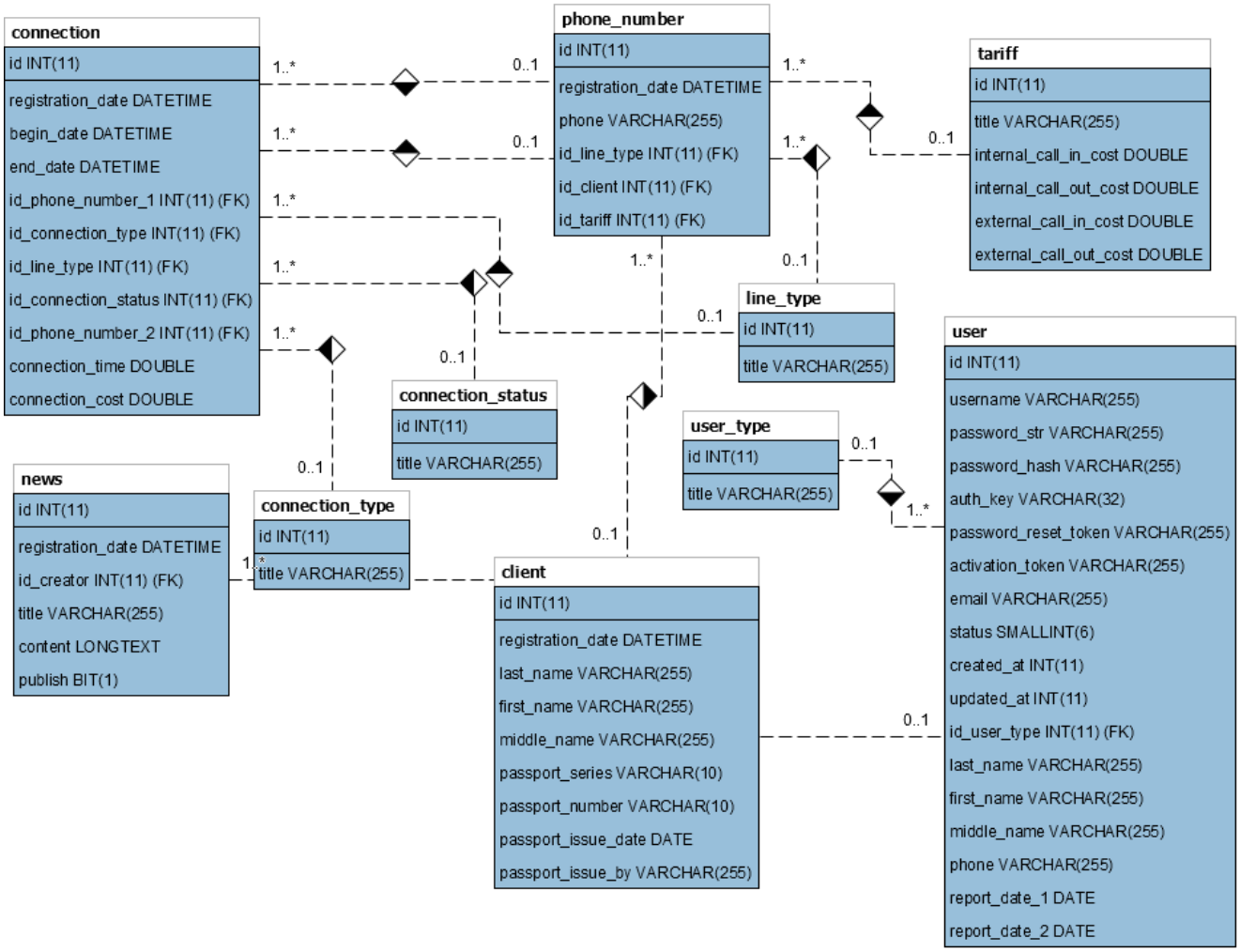

Рисунок 2.12 – Физическая модель базы данных

Описание таблиц базы данных приведено ниже.

Таблица 2.1 – Структура таблицы client

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
registration_date	datetime	Да	NULL
last_name	varchar(255)	Да	NULL
first_name	varchar(255)	Да	NULL
middle_name	varchar(255)	Да	NULL
passport_series	varchar(10)	Да	NULL
passport_number	varchar(10)	Да	NULL
passport_issue_date	date	Да	NULL
passport_issue_by	varchar(255)	Да	NULL

Таблица 2.2 – Структура таблицы connection

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
registration_date	datetime	Да	NULL
begin_date	datetime	Да	NULL
end_date	datetime	Да	NULL
id_phone_number_1	int(11)	Да	NULL
id_connection_type	int(11)	Да	NULL
id_line_type	int(11)	Да	NULL
id_connection_status	int(11)	Да	NULL
id_phone_number_2	int(11)	Да	NULL
connection_time	double	Да	NULL
connection_cost	double	Да	NULL

Таблица 2.3 – Структура таблицы connection_status

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
title	varchar(255)	Да	NULL

Таблица 2.4 – Структура таблицы connection_type

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
title	varchar(255)	Да	NULL

Таблица 2.5 – Структура таблицы line_type

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
title	varchar(255)	Да	NULL

Таблица 2.6 – Структура таблицы news

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
registration_date	datetime	Да	NULL
id_creator	int(11)	Да	NULL
title	varchar(255)	Да	NULL
content	longtext	Да	NULL
publish	bit(1)	Да	NULL

Таблица 2.7 – Структура таблицы phone_number

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
registration_date	datetime	Да	NULL
phone	varchar(255)	Да	NULL

id_line_type	int(11)	Да	NULL
id_client	int(11)	Да	NULL
id_tariff	int(11)	Да	NULL

Таблица 2.8 – Структура таблицы tariff

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
title	varchar(255)	Да	NULL
internal_call_in_cost	double	Да	NULL
internal_call_out_cost	double	Да	NULL
external_call_in_cost	double	Да	NULL
external_call_out_cost	double	Да	NULL

Таблица 2.9 – Структура таблицы user

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
username	varchar(255)	Нет	
password_str	varchar(255)	Да	NULL
password_hash	varchar(255)	Нет	
auth_key	varchar(32)	Нет	
password_reset_token	varchar(255)	Да	NULL
activation_token	varchar(255)	Да	NULL
email	varchar(255)	Нет	
status	smallint(6)	Нет	10
created_at	int(11)	Нет	
updated_at	int(11)	Нет	
id_user_type	int(11)	Да	NULL
last_name	varchar(255)	Да	NULL
first_name	varchar(255)	Да	NULL
middle_name	varchar(255)	Да	NULL

Окончание таблицы 2.9

Поле	Тип	Null	По умолчанию
report_date_1	date	Да	NULL
report_date_2	date	Да	NULL

Таблица 2.10 – Структура таблицы user_type

Поле	Тип	Null	По умолчанию
id	int(11)	Нет	
title	varchar(255)	Да	NULL

2.3 Обоснование проектных решений

2.3.1 Выбор и обоснование технических решений

Под техническим обеспечением понимается вся совокупность средств, позволяющих создать техническую основу для использования информационных систем, то есть сетевое и кабельное оборудование, серверное оборудование, персональные компьютеры пользователей, принтеры, сканеры и другую оргтехнику.

Разрабатываемый программный продукт имеет клиент-серверную архитектуру.

Архитектура клиент-сервер основана на распределении функций между двумя типами независимых и автономных процессов: серверами и клиентами. Сеть связывает воедино серверы и клиенты, предоставляя средства связи.

В архитектуре клиент-сервер серверная часть базы данных не только отвечает за доступ к общим данным, но и анализирует их. Клиент передает на сервер запросы на чтение или корректировку данных, которые построены на языке SQL. Сервер сам реализует требуемые выборки или коррективы, отслеживая в любом процессе полноценность и нерушимость данных, а итоговый параметр кода возврата или набора записей передает на персональном компьютере клиента.

Минусы файлового сервера выражены в основном тем, что данные находятся в одном месте, а анализируются в другом. Они должны проходить по сети, что зачастую резко нагружает сеть и, как результат, снижает быстродействие программного обеспечения при росте числа работающих сразу клиентов.

Также к недостаткам самой архитектуры сервера можно отнести децентрализованное решение реализуемых проблем полноценность и неизменности данных, а также их доступности. Такое решение явно минимизирует стабильность приложения.

Клиент-серверная архитектура сводит на нет все указанные недостатки. Т.к. она позволяет лучшим образом разделять нагрузки между клиентом и сервером, что оказывает явное влияние на системы в целом: поддержку, цену и скорость работы.

К примеру, подобная архитектура помогает поддержать дополнительную надежность и стабильность системы, роста мощности системы и числа активных подключений за счет применения нескольких серверных приложений. Также доступен перенос базы данных с одной СУБД на другую и корректировка требуемых данных без частого обновления клиентских программ. Сервер приложений еще и явно минимизирует нагрузку на сервер базы данных.

Самый правильный шаг для решения выбранной задачи становится внедрение архитектуры тонкого клиента и толстого сервера, т.к. 3 звена в архитектуре не требуются для текущей задачи, а цена и сложность реализации подобной сложной архитектуры явно больше, чем 2-уровневой. Да и при том, текущая информационная инфраструктура без проблем подойдет только для 2-уровневой архитектуры.

Для корректного взаимодействия компонентов клиент-серверной архитектуры между собой требуется их соответствие некоторым основным правилам. Эти правила должны в равной степени выполнять и клиенты, и серверы, и прикладное программное обеспечение.

Эксплуатация разрабатываемой системы должна производиться на рабочих станциях следующей конфигурации:

– операционная система — Windows;

– процессор — Pentium/Celeron/Athlon 1000-1800 МГц;

– память (ОЗУ) — 1024 Мб;

– жесткий диск — не менее 200 МБ свободного места (для установки программного обеспечения);

– монитор — позволяющий работать с разрешением экрана не менее 1152x864 пикселей;

– устройства ввода информации (клавиатура, мышь);

– дополнительное программное обеспечение — Oracle Java SE Runtime Environment (JRE) 7.

Однако для сервера баз данных понадобится более мощное техническое обеспечение:

– операционная система — FreeBSD, Linux, Mac OS X, Solaris, Windows Server;

– процессор — Pentium/Celeron/Athlon 2400-3000 МГц;

– память (ОЗУ) — 2024-3072 Мб;

– жесткий диск — определяется объемом базы данных;

– устройства ввода информации (клавиатура, мышь).

2.3.2 Выбор и обоснование инструментальных средств программирования

Далее важно определить язык программирования, который поможет создать приложение для взаимодействия с базой данных.

При формировании программного комплекса следует рассматривать его как отложную систему, разные части которой взаимодействуют друг с другом для решения общей задачи в то же время, различные его программные модули могут

решать независимые задачи При проектировании должны программные систем необходимо поддерживать такого рода модульность, как для разделения труда программистов, так и доя упрощения поддержки конечного продукта и расширения его функционала.

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

– кроссплатформенность;

– торговая мультипарадигменность;

– эффективность;

– безопасность;

– выразительность;

– поддержка;

– расширяемость;

– поддержка методов параллельного программирования.

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

означает наличие компилятора или среды выполнения под различные аппаратные и программные платформы.

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

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

Эффективность.

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

Компилируемые языки.

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

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

Языки с компиляцией в байт-код.

Такие языки транслируются в промежуточное представление -– байт код виртуальной машины, который можно рассматривать как аналог ассемблера доя целевой виртуальной машины. Как правдой содержат средства JIT(англ. Just-in-

Time, компиляция «на лету») компиляции, что почти уравнивает их по скорости выполнения с компилируемыми языками.

Интерпретируемые языки.

Программный код написанной на этот категории языков работает медленнее всего. Причина в том, что в случае компиляции в машинного кода из байт-кода компилятор строит абстрактное синтаксическое дерево на основе всей информации о программе, которое анализируется и затем оптимизируется. Интерпретатор лишен возможности проанализировать весь программный код целиком, он вынужден анализировать команды построчно и интерпретировать их, каждую в отдельности.

Безопасность.

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

Выразительность языка.

Язык накладывает множество ограничений при проектировании архитектуры программного обеспечения. Языковые конструкции одновременно ограничивают программиста и в то же время предоставляют ему некоторый уровень абстракции. Так, например, программы, написанные в оперативном стиле на ассемблере предоставляют неограниченный контроль над машиной, но в то же время, код на языке ассемблера труден доя восприятия человеком, его сложно

модифицировать и поддерживать. Это происходит по причине того, что ассемблерные инструкции описывают элементарные операции над данными, в то время как программист представляет логику работы более абстрактно.

За структурным программированием появились объектно-ориентированное и др. При всем этом разнообразии рассматриваемые далее языки за исключением С, относятся к объектно-ориентированному типу но при этом поддерживают и оперативную и функциональную парадигмы, однако большинство кода на такта языках пишется в объектно-ориентированном стиле. Объектно-ориентированная парадигма прочно задала свое место в разработке программных систем благодаря основным принципам: наследованию, инкапсуляции, полиморфизму и абстрагированию. Так же как все структурные языки имеют реализации цикла, ветвления, так и все объектно-ориентированные языки так или иначе реализуют языковые средства для организации сокрытия функционала за некоторым интерфейсом, полиморфного поведения объектов языке и обеспечения низкой связанности компонентов. Однако все языки делают это по- своему.

В настоящее время программисты не испытывают недостатка в выборе средств программирования: существует множество различных языков, как универсальных, используемых во многих сферах программирования и способных работать на нескольких аппаратных и программных платформах, так и специализированные, ориентированные на определенную область разработки, как, например, PROLOG – язык программирования высокого уровня, предназначенный для реализации систем и программ искусственного интеллекта. Существует множество различных классификаций языков программирования: по уровню семантики (языки высокого и низкого уровня), по используемым парадигмам программирования, по безопасности и по ряду других характеристик. Совокупность требований к разрабатываемой системе и определяет выбор средств ее реализации на этапе проектирования; учитываются требования к ее быстродействию, надежности, информационной безопасности.

Цель данной работы – провести сравнительный анализ свойств языков Python и PHP. Условно можно разделить эту работу на две основные части:

- теоретическая часть, содержащая краткий обзор возможностей языков программирования и сферы их применимости;
- практическая часть, содержащая данные о проведенном эксперименте с приведением кода программ, выполняющих аналогичные действия, написанных на исследуемых языках, выходные данные и их непосредственный анализ, включающий сравнительную характеристику времени выполнения программы, объем исполняемых файлов и нагрузку на память ЭВМ.

В результате по полученным данным можно сделать обобщенный вывод о характеристиках языков и степени их различия.

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

Также Python обладает развитой системой модулей, как стандартных, также написанных на Python, а, следовательно, обладающих теми же преимуществами, такими как кроссплатформенность, и позволяющих реализовывать наиболее общие задачи, так и специфических, применяемых в том случае, когда необходимо решать более широкий круг задач.

Ввиду широкого ряда функциональных возможностей Python может использоваться в самых различных сферах; с его помощью можно осуществлять разработку web-приложений, автоматизированных информационных систем, научных вычислительных комплексов, графических пакетов.

PHP – язык программирования общего назначения, созданный преимущественно для работы над web-приложениями, для генерации HTML-страниц и работы с базами данных. PHP обладает Си-подобным синтаксисом, также является парадигмальным и кроссплатформенным, ядро PHP также реализует средства для автоматического управления памятью; вся выделенная память возвращается системе после завершения работы скрипта [2].

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

Наиболее широкое применение PHP находит при разработке web-приложений: подавляющее большинство сайтов и сервисов разработано именно с его помощью. Множество различных web-фреймворков и CMS-систем имеет в своей основе именно PHP.

Итак, Python и PHP в целом имеют схожие исходные характеристики. Необходимо рассмотреть их возможности при реализации конкретной задачи.

По быстродействию программа, написанная на Python, уступает аналогичной на PHP в 3,5 раза, однако, загружает память в 1,5 раза меньше. Необходимо подчеркнуть, что приведенные данные носят приближенный характер ввиду возможных погрешностей измерения характеристик при использовании методов с ограниченной точностью. Тем не менее они позволяют дать общее представление о производительности кода и в соответствии с поставленными задачами и требованиями, предъявляемыми к ним, сделать выбор в сторону того или иного языка. Можно также выделить, что PHP оправданно широко используется в web-программировании, учитывая скорость выполнения его скриптов. В целом полученные результаты соответствуют известным раннее выводам о характеристиках данных языков.

Следовательно, для реализации рассмотренной информационной системы выбираем язык программирования PHP.

Основным языком веб программирования становится PHP. Его главными преимуществами считаются: понятный синтаксис, хорошее быстродействие, работа с большинством хостингов. Также плюсом будет то, что на PHP созданы многие используемые движки.

Другой известный язык веб программирования на платформе Unix -– язык Perl. Он сложнее, его синтаксис запутаннее, и изначально не был создан для для веб программирования.

Потому в рамках языка разработки соединения с базой данных выбираем PHP.

PHP был выбран, т.к.:

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

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

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

Сейчас реляционная модель данных становится базовой при построении базы данных. Это получилось из-за того, что организация хранения данных в виде таблиц логична для пользователя. Эта особенность приводит к множеству реляционных СУБД. Их параметры значительно разнятся. Подбор самой адекватной СУБД стал сложной задачей.

Опишем самые распространенные СУБД.

1. Oracle Database.

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

– Enterprise Edition: Полноценная версия без ограничений. Эта корпоративная редакция продукта нужна для крупных предприятий. Пользователям даются опции, которые помогают архитектурно и функционально оптимизировать сервер;

– Standard Edition: Есть ограничение по числу процессорных разъемов (не более 4-х). Редакция нужна для применения в компании среднего размера или в отделе крупной компании;

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

– Personal Edition: Версия рассчитана на однопользовательский режим;

– Lite: Версия используется в мобильных и встраиваемых устройствах. Также часто применяется в малых предприятиях;

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

Все эти 6 версий СУБД имеют идентичный исходный код и аналогичный функционал, исключая отдельные узконаправленные методики конкретных версий. Главная задача стандартной, персональной и мобильной версий – понижение стоимости владения, легкость использования ПО.

Oracle Database считается кроссплатформенным ПО. Это реализовано благодаря тому, что около 80 % программного кода написано на языке Си. А ядро сервера, которое имеет остальные 20 % кода, переделывается под необходимую платформу, поскольку создано на машинно-зависимых языках [14].

2. Microsoft Access.

Реляционная СУБД, созданная Microsoft. Имеет встроенный Visual Basic for Applications (VBA), позволяющий создавать приложения в Access для работы с

базой данных. Понимо VBA в приложении применяется язык структурированных запросов SQL и макрокоманды [15].

MS Access считается файл-серверным СУБД, что уменьшает круг её использования. В роли движка базы данных стоит Access Database Engine или Microsoft Jet 4.0 в зависимости от ревизии СУБД [1]. MS Access может совмещаться с внешними СУБД клиент-серверной архитектуры, например, с MySQL, Firebird, Oracle и др. Устойчив к сбоям в электропитании благодаря автоматическому резервированию после перехода к следующей записи. Программный комплекс лучше всего применять после приобретения лицензии, хотя есть версии, доступные открыто.

Проект MS Access сохранен в файле формата accdb, упрощая тем самым его передачу и работу с программой. Различные конструкторы помогают взаимодействовать с данной СУБД персоналу, которые имеют недостаточный уровень знаний. Плюсом такой настольной СУБД становятся русифицированный интерфейс и хорошие средства защиты информации.

3. MS SQL Server.

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

Microsoft SQL Server Express Edition — это версия продукта, уменьшенного по функционалу, но находящаяся в открытом доступе [18]. Эта версия применима лишь в рамках малой компании в силу уменьшенных возможностей по сравнению с MS SQL Server. Языком просмотра выступает процедурное расширение языка SQL — Transact-SQL.

Microsoft SQL Server совместим с базами данных других форматов, к примеру: Oracle, DB2, Sybase и Microsoft Access [19]. Эта СУБД имеет простой

доступ пользователей к анизучаемой информации, что реализовано методов объединения с пакетом программ Microsoft Office. Также имеется возможность шифровать базу данных, файлы журналов или файлы данных, тем самым реализуя дополнительную защиту данных.

4. Sybase A daptive Server Enterprise.

Реляционная СУБД, создана фирмой SAP [20]. Сначала была разработан вместе с Microsoft SQL Server, и потому тоже применяет Transact-SQL как базовый язык запросов. Является СУБД клиент-серверной архитектуры. Используется в системах масштаба среднего или большого предприятий. Определяется удобством использования и минимальной ценой для сегмента рынка СУБД, поддерживающих большие базы данных. Применяется при необходимости взаимодействия с большим числом пользователей, большими объемами данных для очень важных объектов. Также данная СУБД не зависит от программного обеспечения других производителей.

SAP Sybase Adaptive Server Enterprise Cluster Edition — версия СУБД, имеющая усиленную отказоустойчивость, обеспечиваемую благодаря разбиению на кластеры и перевода пользователей с отказавшего узла кластера на рабочий [21]. При сбое в процессе реализации транзакции операция выполнится повторно по факту перехода на рабочий узел.

5. ЛИНТЕР.

СУБД российского производства, созданная научно-производственным предприятием РЕЛЭК (Реляционные экспертные системы). Считается кроссплатформенным программным обеспечением, поддерживающим почти все операционные системы. Базовыми направлениями использования становятся гос. проекты, встроенные системы и онлайн-системы. СУБД ЛИНТЕР имеет отличную надежность благодаря системе оперативного резервирования, другими словами – резервная система приходит на замену вышедшей из строя в авто режиме. Язык запросов в СУБД ЛИНТЕР отвечает требованиям SQL:2003 [22].

Существует 4 версии СУБД: ЛИНТЕР Бастион, ЛИНТЕР Стандарт, ЛИНТЕР Realtime и ЛИНТЕР Multiversion.

6. MySQL.

Универсальная СУБД. Лежит в свободном доступе, но имеет коммерческую лицензированную версию от MySQL AB. СУБД с открытым кодом [12]. Изначально создавалась шведской компанией MySQL AB, с 2008 по 2010 года — уже фирмой Sun Microsystems. Сейчас разработка и поддержка программного обеспечения лежит на Oracle. Первая версия СУБД вышла в 1995 году, а представлена она была в январе 2001 года. Данная СУБД используется для управления малых и средних систем. СУБД MySQL используется там, где есть клиент-серверная и встроенная архитектура. В версиях выше MySQL 3.22 все ограничения с таблиц сняты [23].

Есть и визуальный интерфейс для упрощения работы с базами данных MySQL — PHP MyAdmin [11]. СУБД MySQL версий 5.0 и выше отвечает стандарту структурированного языка запросов SQL, поэтому она может быть совмещена с многими версиями БД. Главный язык разработки – C/C++.

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

2.4 Тестирование ИС

Экспертизу качества модуля реализуют на фазах жизненного цикла в рамках ГОСТ 28195-89 «Оценка качества программных средств» . Состоит она из определения номенклатуры параметров, их оценку и сравнение значений, полученных по итогу сравнения с исходными показателями. Эти параметры качества соединены в систему из 4 уровней, описанных ниже. Допускается использовать дополнительные показатели на любом уровне.

Для поддержания доступности реализации интегральной оценки по группам показателей качества применяют факторы качества (уровень 1): стабильность

программных средств, настройка, удобство применения, результативность, универсальность, скорость работы.

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

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

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

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

Тестирование является важной и обязательной частью процесса разработки. Причем, в разработке программы необходимо не только финальное, обязательное тестирование, но и периодичное.

В ходе контроля программы должны быть проведены следующие виды тестирования:

1. Тестирование в нормальных условиях
2. Тестирование в исключительных условиях, в том числе:

a. Тестирование нулевыми данными.

b. Тестирование чужими данными.

с. Тестирование избыточными данными.

3. Оценка полноты проверки программы.

Начальные испытания нужно проводить для отражения работоспособности программы, качественных и количественных параметров, важности изменения документации и частей программы, оптимизации последующих этапов разработки.

Перед проведением испытания программы разработчик должен подготовить всю необходимую документацию в виде документа в формате .pdf.

При этом в данной документации должны быть описаны все характерные особенности использования программного обеспечения и варианты работы с ними.

Перед проведением испытания программное обеспечение должно быть установлено на сервер заказчика и должно быть готово к использованию, в том числе должны быть введены тестовые данные, разработанные с участием заказчика на основании реальных данных.

Далее разработчик демонстрирует работы информационной системы, включая все обязательные функции, такие как регистрация пользователя, его авторизация, восстановление доступа, ввода данных в систему, их удаление и редактирование, формирование расписания и дальнейшую работу с ним. При этом в обязательном порядке должны быть изучены ситуации работы программного обеспечения при удалении части данных, а также преднамеренном вводе информации в неверном формате и реакция системы на такие ситуации.

При обнаружении каких-либо ошибок в работе программного обеспечения они должны быть задокументированы, а заказчик и разработчик должны однозначно понимать род ошибки и результаты ее воздействия на систему. При этом в ходе диалога разработчика и заказчика должен быть установлен срок исправления ошибок. После их устранения процедура выполняется еще раз в количестве итераций, необходимых для обеспечения полностью безошибочной работы программы. После достижения такого результата разработчиков и

заказчиком подписывается акт об отсутствии ошибок в программном обеспечении и приеме информационной системы заказчиком.

Прикладное программное обеспечение будет тестироваться на локальной машине с использование сервера. Будет применяться комплексное тестирование имитирующее работу пользователя и администратора с системой. В ходе тестирования будут выполнены следующие действия: 1) регистрация нового изделия, 2) внесение данных в справочники, 3) добавление изделий и проверок, 4) анализ результатов проверок, 5) формирование отчетов.

Тестирование чужими данными.

Тестирование чужими данными проводилось в некоторых экранных формах и при авторизации пользователя.

Например, в поле, имеющее числовой тип, нет возможности внести данные типа строка.

Тестирование избыточными данными.

Исключительные данные представляют собой ввод данных в избыточном объеме. Программа определяет такие ситуации, выводит сообщение, что в данной ячейке вводится ограниченное количество знаков. Например, при добавлении телефона пользователя с количеством цифр более 11 программа выдаст ошибку.

Оценка полноты проверки программы.

В ходе тестирования система эксплуатировалась в условиях, близких к реальным.

Система не позволяет пользователю вводить большинство недопустимых значений, что существенно упрощает процесс тестирования, минимизирует риск возникновения исключительных ситуаций и позволяет достаточно полно проверить работоспособность программного обеспечения в нормальных и экстремальных условиях. Вывод сформированных отчетных форм на печать работает без сбоев.

2.5 Разработка документации на систему

Программирование является процессом создания программы, и при классической ручной разработке программного продукта без привязки к методу проектирования может представляться как череда следующих шагов:

– выявление требований к программному обеспечению;

– выбор или создание алгоритма решения поставленной задачи;

– подготовка команд;

– проверка;

– тестовые запуски.

Выявление требований к программному обеспечению – важный этап, где подробно указывается начальная информация и составляются требования к итогу работы. Также определяется поведение программы в разных случаях.

В процессе выбора или создания алгоритма важно определить череду действий, необходимых для достижения результата. Многие задачи часто решаются различными способами. В этом случае разработчик, применяя отдельный критерий, к примеру, быстроту работы алгоритма или его точность, находит оптимальное решение. Далее происходит подробное описание алгоритма. По факту выделения требований к программе и подготовки алгоритма решения, его записывают на подходящем языке программирования.

Проверка программного обеспечения заключена в нахождении и корректировке ошибок, которые были совершены на этапах подготовки задачи для реализации на ЭВМ.

Ошибки делят на синтаксические и семантические.

Синтаксические ошибки, связанные с неправильным написание операторов языка программирования, находятся на этапе трансляции программы. Тут даже можно увидеть сообщение, описывающее оператор, в котором допущена ошибка, место, где она обнаружена, а также текст, поясняющий суть проблемы.

Семантические ошибки, связанные, зачастую, с неправильным построением математической модели и алгоритма выполнения задачи, при процедуре прохода программы сразу не обнаруживаются, поскольку программа выполняется и даже выдает итоговый результат.

Для сверки корректности программы выполняется ее тестирование, т.е. ее реализация при таких исходных данных, для которых итоги решения уже известны. При подборе тестовых наборов исходных данных важно учесть доступность проверки разных вариантов решения задачи. Оценка корректности программы реализуется методом сравнения результатов, полученных при реализации программы, с имеющимися тестовыми.

Тестирование помогает доказать факт наличия смысловой ошибки. Для ее определения и выявления причины, реализовавшей ошибку, лучше всего применять отладочные операторы, позволяющие контролировать проход программы, следить за чередой выполнения операторов программы и корректировкой значений элементов массивов и переменных в процессе работы программы, отследить правильность индексирования элементов массивов, отслеживать обращение к подпрограммам.

Тут же важно проверить, как работает программа на максимальном количестве входных наборов данных, пусть даже не самых верных. Опишем более подробно основной шаг процесса программирования – алгоритмизацию.

Задача программиста заключена в продумывании последовательности действий, выполнив которые, можно достигнуть требуемого результата.

Под алгоритмом, как правило, подразумевается строго описанный порядок действий для выполнения процесса, имеющего сформулированную цель.

Алгоритм должен иметь следующие четкие свойства:

– строгая и однозначная определенность, которая не позволяет понять алгоритм по-другому;

– универсальность, то есть алгоритм можно применять на любых языках программирования и в любой среде;

– продуктивность, то есть прежде всего должен быть достигнут результат действия алгоритма.

Опишем порядок использования системы.

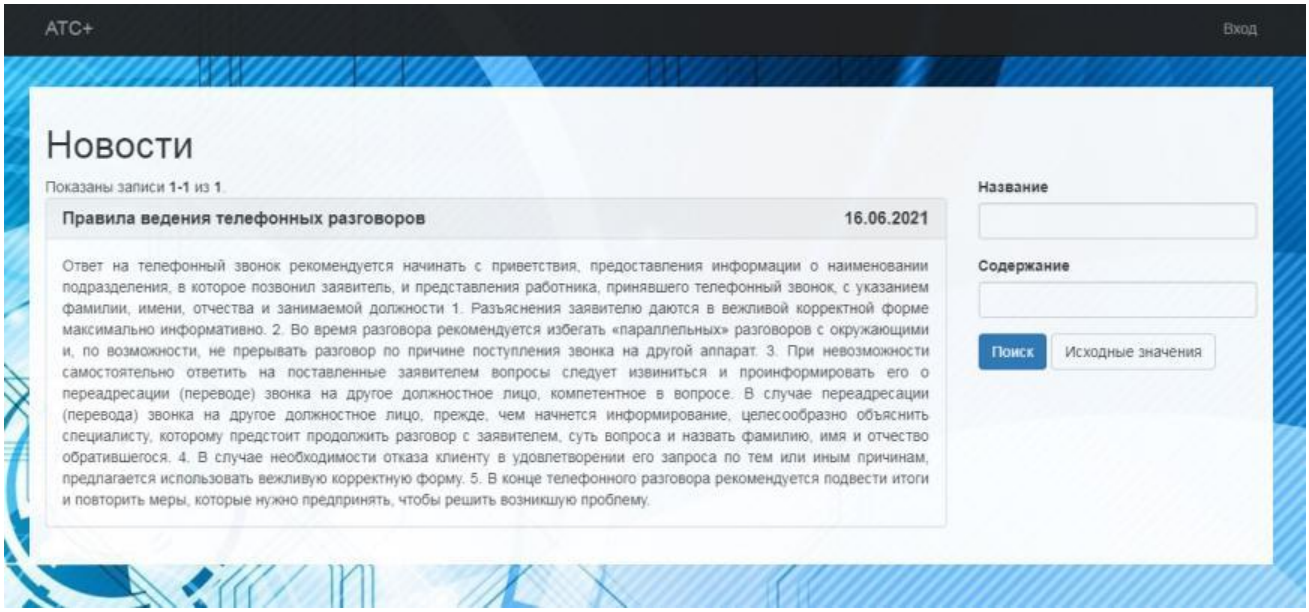

Рисунок 2.13 – Главная форма программы

На главной форме программы приведен список новостей. Для перехода в панель управления пользователь должен авторизироваться.

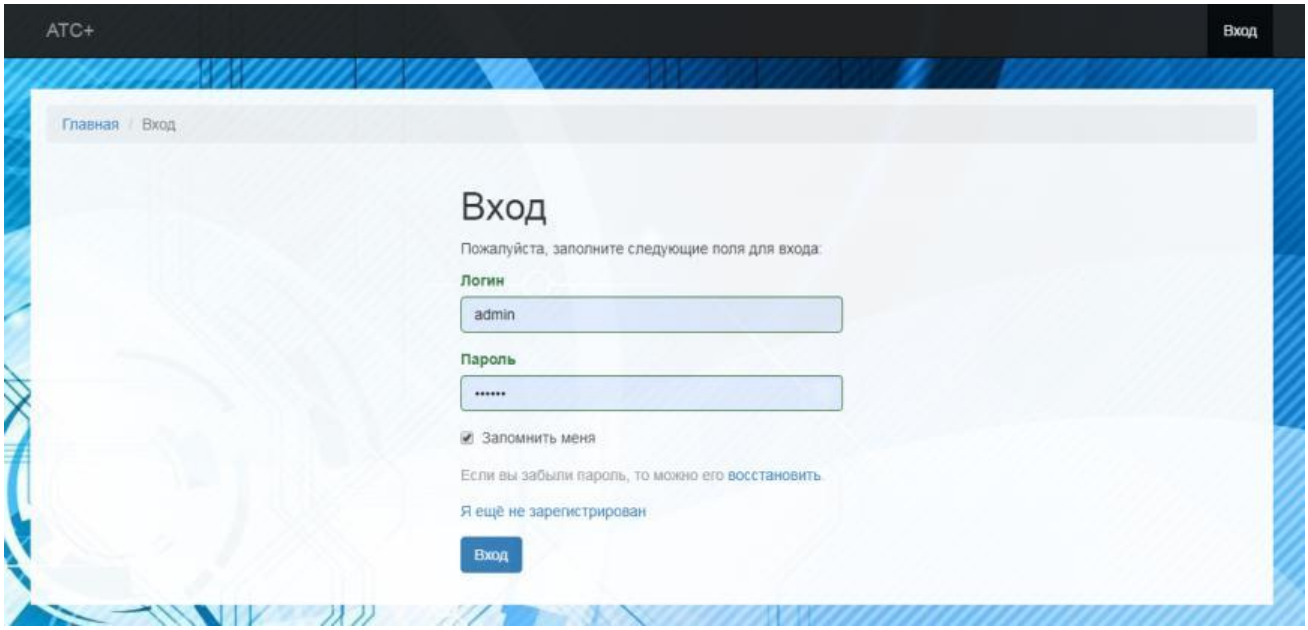

Рисунок 2.14 – Страница авторизации

Далее администратор приступает к заполнению справочников:

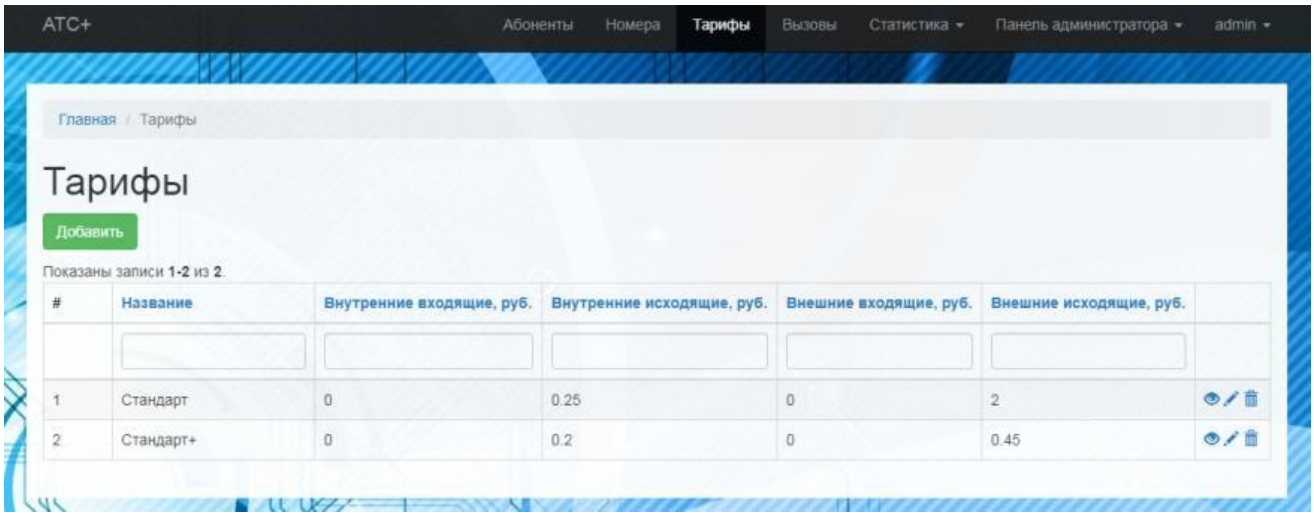

Рисунок 2.15 – Справочник Тарифы

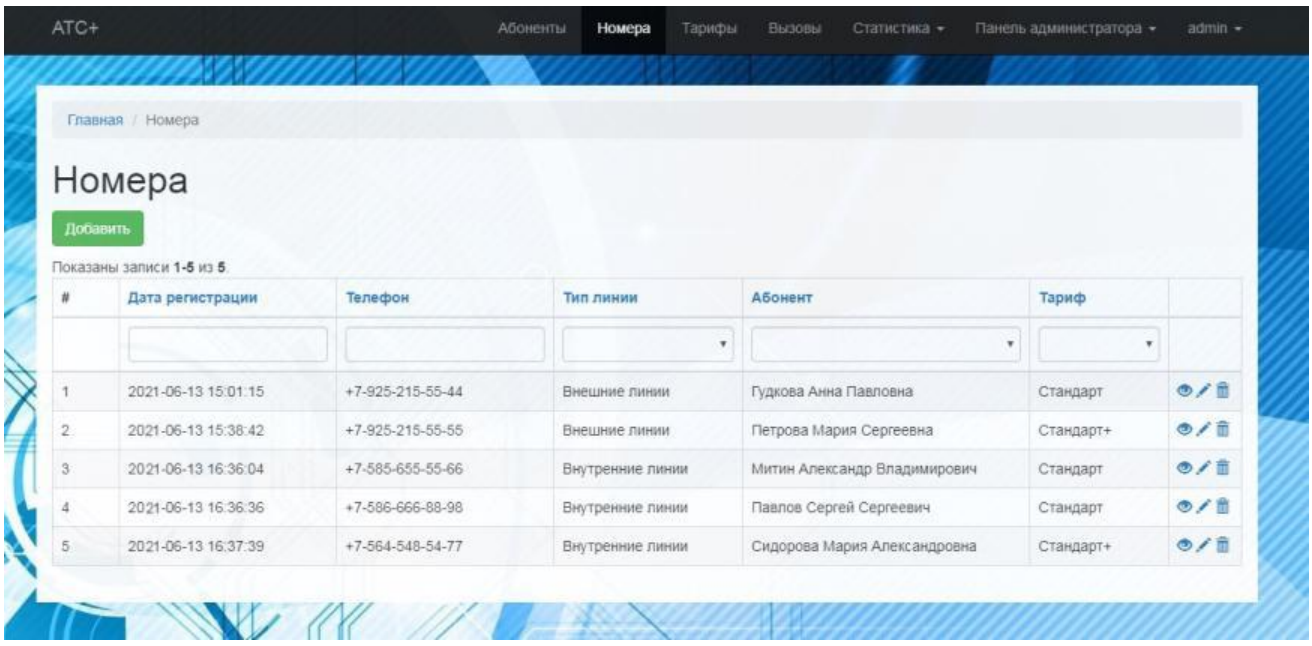

Рисунок 2.16 – Справочник Номера

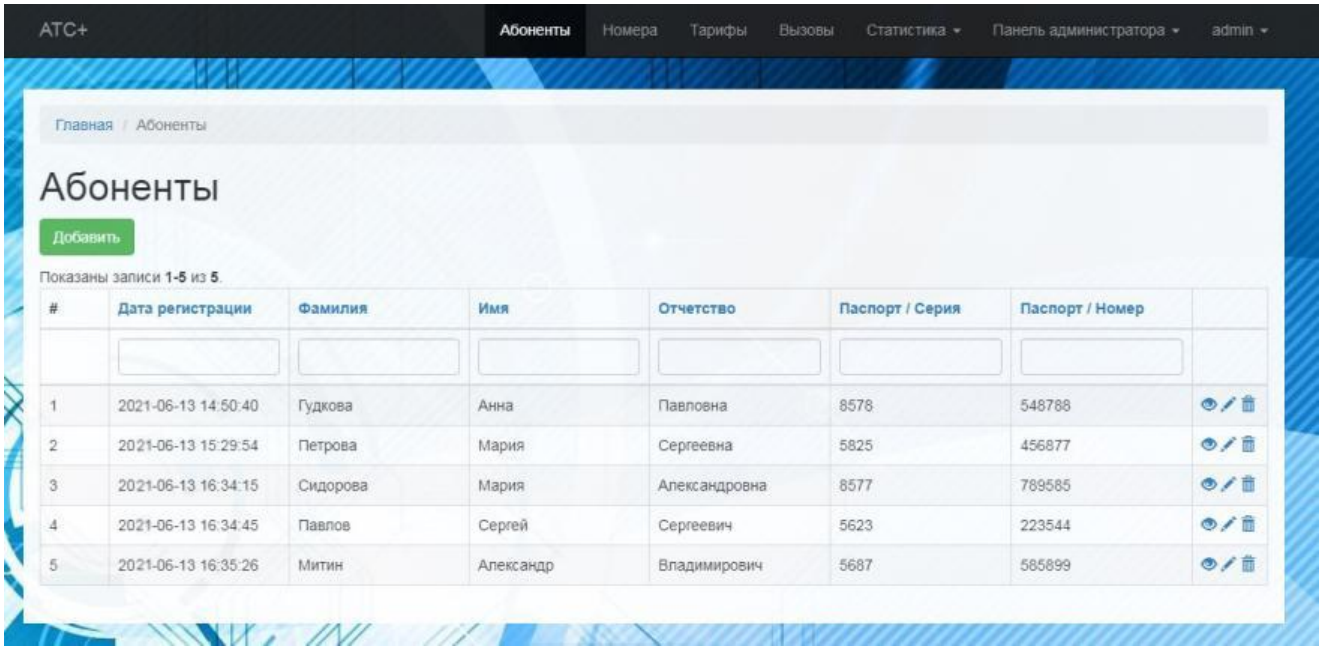

Рисунок 2.17 – Справочник Абоненты

После интеграции с АТС происходит учет телефонных разговоров сотрудников и формируется отчетность.

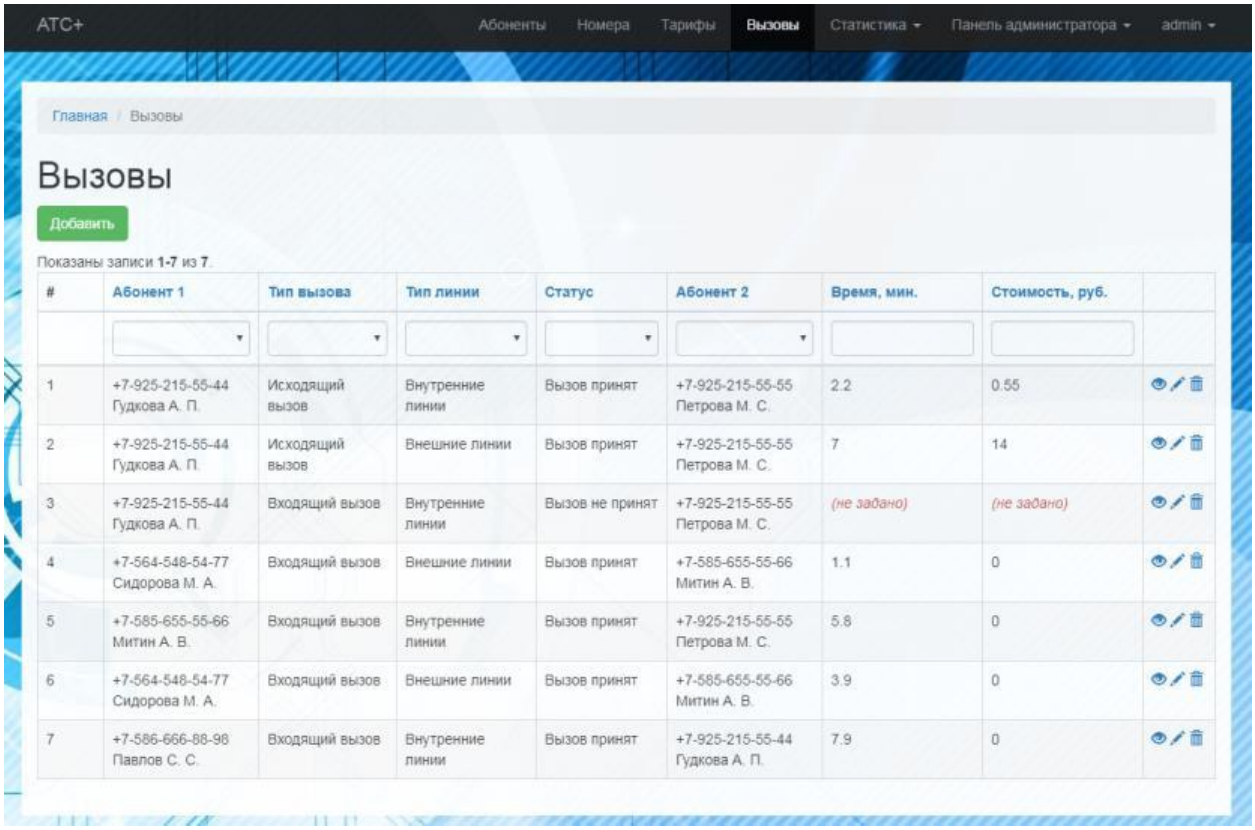

Рисунок 2.18 – Список вызовов

Также формируются отчеты:

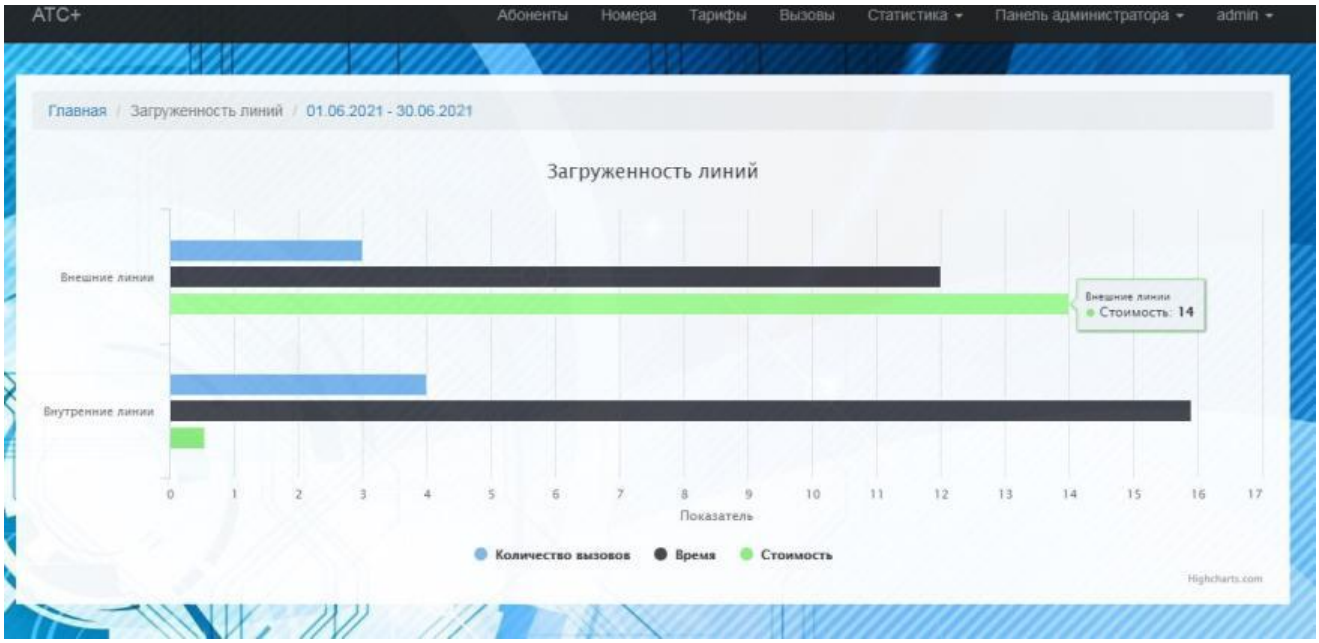

Рисунок 2.19 – Отчет загруженность линий

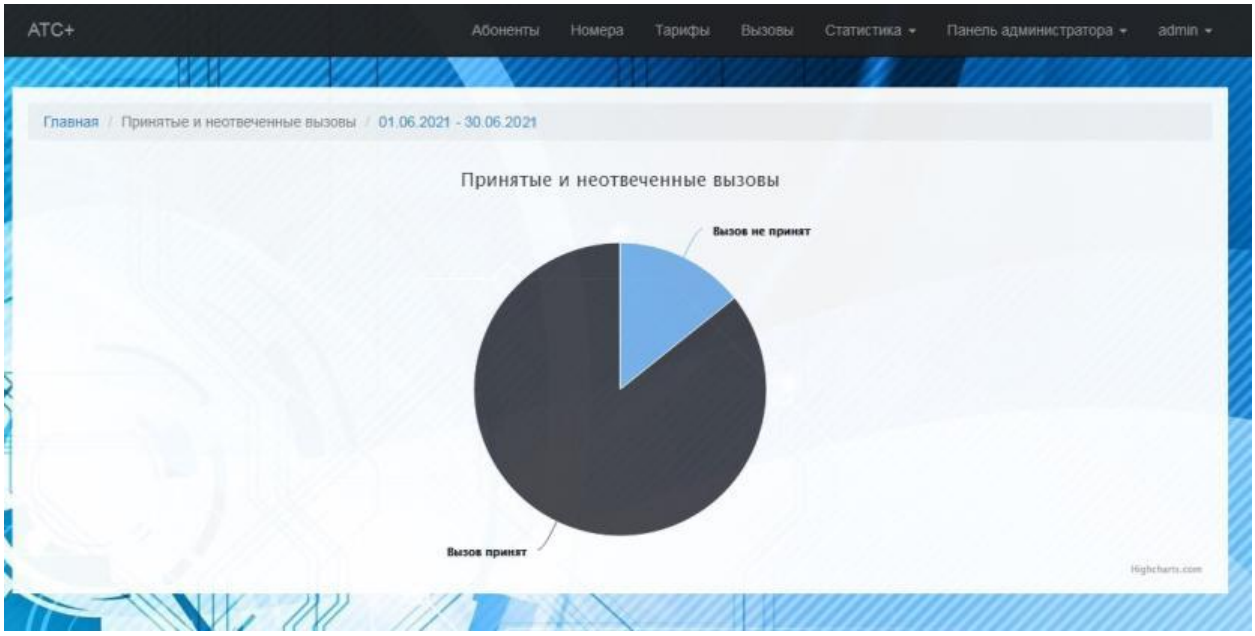

Рисунок 2.20 – Принятые и неотвеченные вызовы

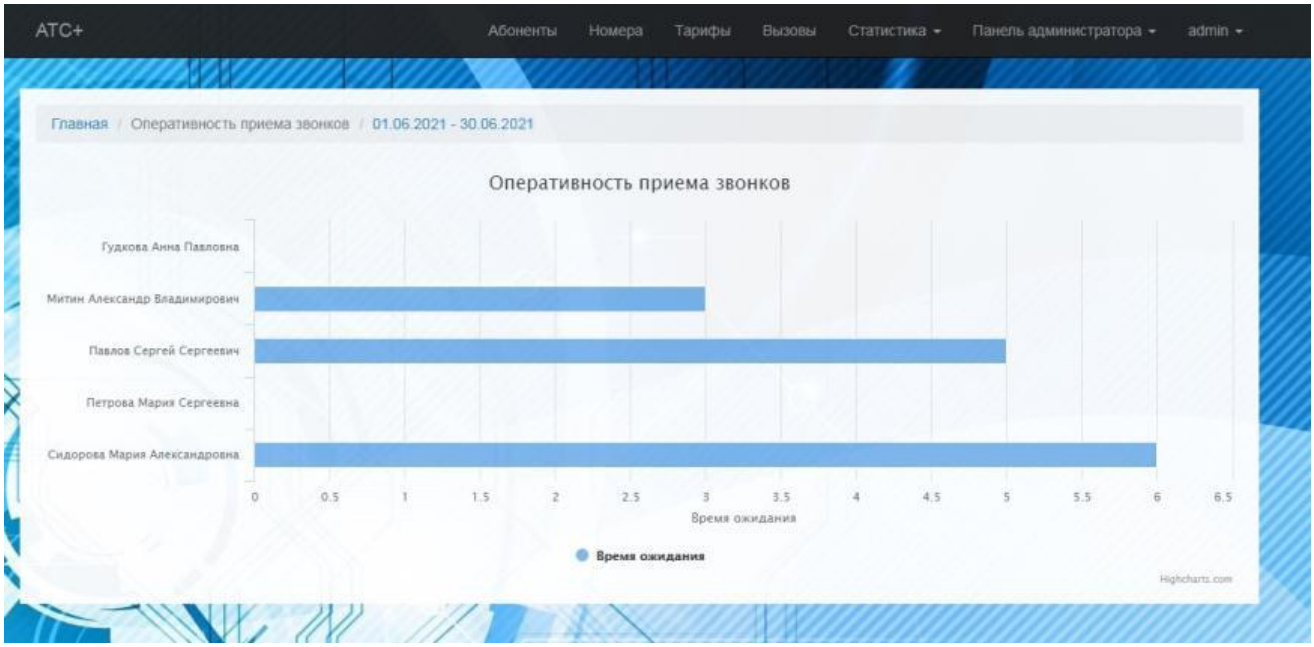

Рисунок 2.21 – Отчет Оперативности приема звонков

Выводы по главе два:

В этой главе разработана структура, подсистемы и объекты информационной базы. Разработан подходящий интерфейс для будущих пользователей. Было произведено тестирование программного продукта.

3 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

3.1 Организационно–экономическая характеристика деятельности предприятия

Ресторан доставки «Суши АНТАНО» осуществляет свою деятельность в сфере общественного питания в Нижневартовске с октября 2019 года.

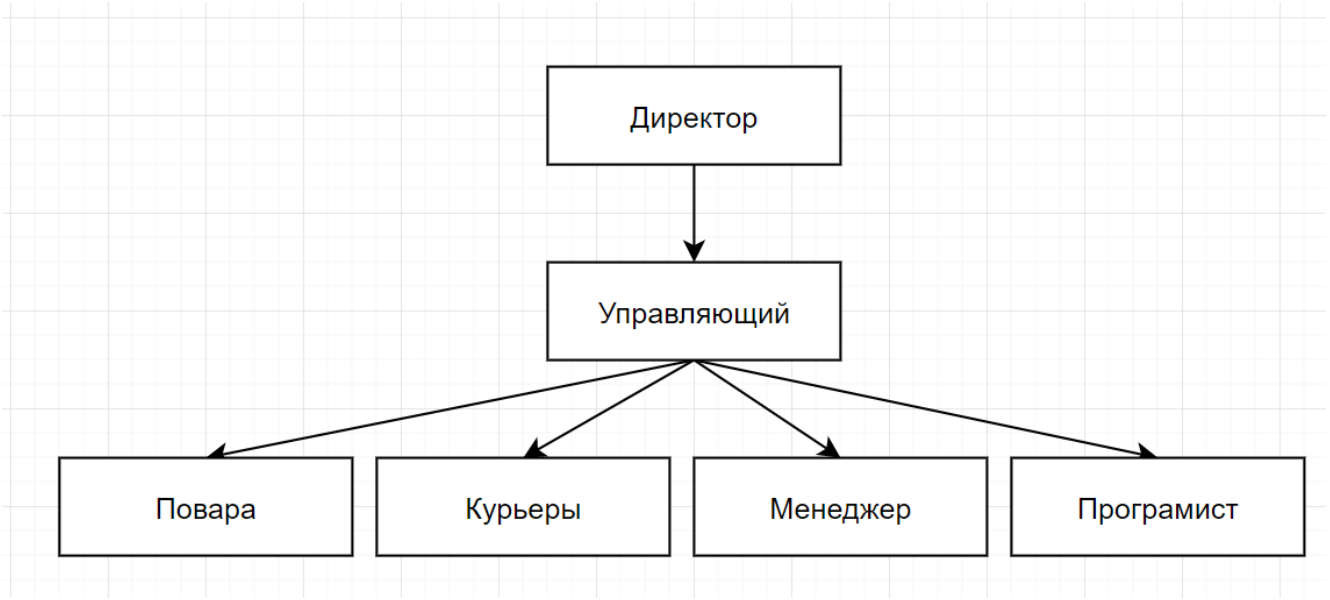

Рисунок 3.1 – Структура управления ресторана доставки «Суши АНТАНО»

3.2 Анализ финансовых показателей деятельности предприятия

В данном пункте рассматривается финансовые показатели ресторана доставки «Суши АНТАНО» за период с марта 2020 года по февраль 2021 года.

Месяц	Выручка общая	Расходы	Доход
Март	552 722	400 722	152 000
Апрель	747 355	653 646	93 709
Май	753 515	575 537	177 978
Июнь	1 025 422	713 281	312 141
Июль	819 414	531 971	287 443
Август	824 305	573 027	251 278
Сентябрь	678 296	642 438	35 858
Октябрь	817 117	545 368	271 749
Ноябрь	710 216	609 662	100 554
Декабрь	784 856	605 478	179 378
Январь	776 896	588 176	188 720
Февраль	894 763	711237	183526

Рисунок 3.2 – Финансовые показатели ресторана доставки «Суши АНТАНО»

Исходя из данной таблицы финансовых показателей ресторана доставки «Суши АНТАНО», данное заведение можно считать рентабельным, заведение за данный период работало без минусовых показателей.

Доход за данный период с марта 2020 года по февраль 2021 года составил 2234334 (два миллиона двести тридцать четыре тысячи триста тридцать четыре) рубля.

В среднем в день приходит от 30 до 65 заказов, средний чек заказов 565 рублей.

3.3 Расчет сметы затрат на реализацию проекта

Для того, чтобы оценить экономическую эффективность разработанной системы, оценим трудовые и стоимостные затраты до и после внедрения автоматизированной системы учета телефонных разговоров сотрудников. При этом будем учитывать среднечасовую заработную плату пользователя системы, которая составляет 267 рублей в час, а также объем используемых в работе системы документов.

Операции технологического процесса при базовом и проектном варианте за год и их характеристики представлены в таблице 3.1 и таблице 3.2.

Таблица 3.1 – Базовый вариант

№ п/п	Наименование операций технологическ. процесса решения комплекса задач	Единица измерения	Объем работы в год	Норма выработки (опер/в час.)	Трудоемкость (часов)	Среднечасовая зарплата специалиста (руб.)	Стоимостн. затраты для ручных операций (руб.)
1.	Учет пользователей	документо строка	3960	75	52,80	201,19	5
2.	Учет телефонов	работа	3960	75	52,80	201,19	5
3.	Учет разговоров	работа	3960	75	52,80	201,19	5
4.	Общий отчет по разговорам	работа	3960	75	52,80	201,19	6
5.	Отчет по	работа	3960	75	52,80	201,19	7

	входящим разговорам						
6.	Отчет по исходящим разговорам	работа	23760	150	158,40	201,19	8
7.	Отчет по разговорам пользователя за период	работа	21120	150	140,80	201,19	9
	Итого:			563,20			117323,01

Таблица 3.2 – Проектный вариант

№ п/п	Наименование операций технологического процесса решения комплекса задач	Оборудование	Ед. Изм.	Объем работы в год	Норма выработки / производительность устройств ЭВМ (опер/в час.)	Трудоемкость (Ч)	Среднечасовая зарплата специалиста (руб.)	Стоимостные затраты (руб.)
1.	Учет пользователей	ЭВМ	работа	3960	155	25,55	201,19	2,5
2.	Учет телефонов	ЭВМ	работа	3960	155	25,55	201,19	2,5
3.	Учет разговоров	ЭВМ	работа	3960	155	25,55	201,19	2,5
4.	Общий отчет по разговорам	ЭВМ	работа	3960	155	25,55	201,19	2,5
5.	Отчет по входящим разговорам	ЭВМ	работа	3960	155	25,55	201,19	2,5
6.	Отчет по исходящим разговорам	ЭВМ	работа	23760	400	59,40	201,19	2,5
7.	Отчет по разговорам за пользователя за период	ЭВМ	работа	21120	250	84,48	201,19	2,5
	Итого:					271,62		

Абсолютный показатель снижения трудовых затрат на обработку информации:

ΔТ = 563,20- 271,62= 291,58 часа

Показатель снижения стоимостных затрат

ΔС = 117323,01- 55326,67= 61996,34 рублей

На рисунке 3.2 приведена диаграмма сравнения базового и проектного вариантов трудовых затрат, на рисунке 3.3 – стоимостных затрат.

Рисунок 3.2 – Диаграмма сравнения базового и проектного варианта трудовых затрат обработки информации.

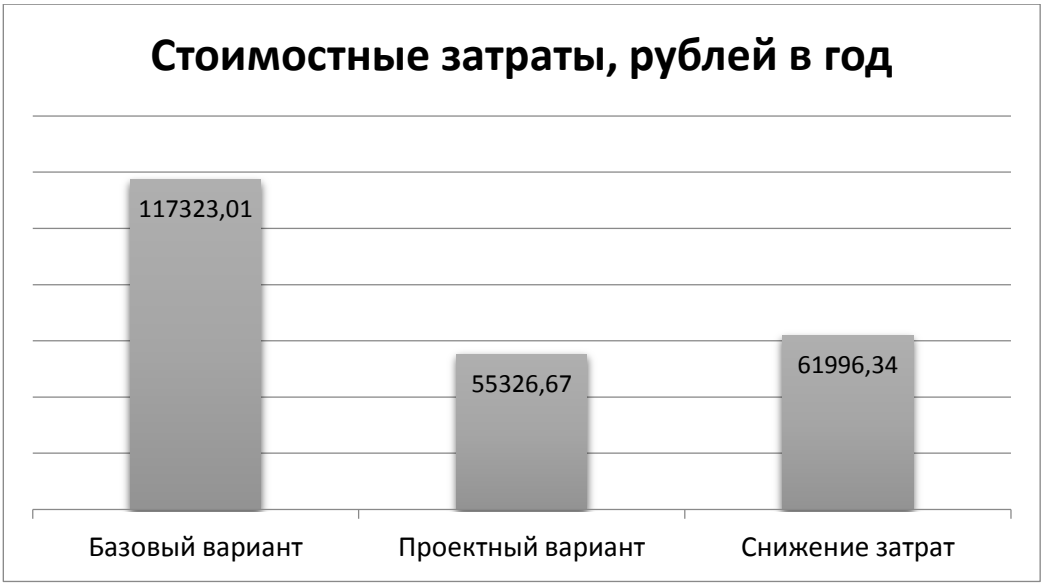

Рисунок 3.3 – Диаграмма сравнения базового и проектного варианта стоимостных затрат обработки информации.

Рассчитаем затраты на внедрение и проектирование системы.

Разработка информационной системы – это трудоемкий процесс, включающий в себя множество этапов: планирование, проектирование, разработку, тестирование, отладку и внедрение созданного программного продукта и прочие. В разработку информационной системы также входят производственные затраты, составляющие себестоимость данного продукта.

Смета затрат на разработку программного продукта включает в себя:

– заработную плату разработчиков;

– отчисления с заработной платы работников (ЕСН);

– эксплуатационные расходы (электричество, отопление);

– расходные материалы;

– фактические затраты, либо затраты будущих периодов (расходы на приобретение средств ЭВМ).

Данные затраты распределяются по этапам разработки программного продукта.

3.4 Определение себестоимости приложения

Целью создания автоматизированной системы учета и планирования продаж является повышение достоверности получаемой информации, а также снижение трудоемкости учета и получения отчетной информации.

Для выполнения работ планируется выделить группу из 2 человек: менеджер проекта (руководитель дипломного проектирования) и студент.

В перечне работ указаны наименования работ и сроки их выполнения. Перечень работ указан в Таблице 3.3.

Таблица 3.3 – Перечень работ

№	Наименование этапа	Исполнитель	Трудоемкость, чел/дн.	Численность, чел.	Длительность, дн.
1.	Анализ требований к системе	Руководитель	6	1	6
2.	Изучение существующих аналогов	Руководитель	6	1	6
3.	Разработка технического задания	Руководитель, студент	36	2	18

Окончание таблицы 3.3

№	Наименование этапа	Исполнитель	Трудоемкость, чел/дн.	Численность, чел.	Длительность, дн.
4.	Постановка задачи	Руководитель, студент	24	2	12
5.	Разработка базы данных	студент	44	2	22
6.	Разработка информационной системы	студент	48	2	24
7.	Разработка интерфейса ИС	студент	48	2	24
8.	Подготовка инструкций пользователям	студент	18	2	9
9.	Подготовка отчета	Руководитель, студент	18	2	9
Итого:					130

Длительность работ составляет 130 дней.

Для расчета экономической эффективности от ИС необходимо определить капитальные вложения в создание и внедрение данного ИС. Капитальные вложения можно разделить на ряд отдельных видов затрат – статей:

– затраты на оплату труда и социальные отчисления;

– накладные расходы и расходы на электроэнергию.

Социальные отчисления рассчитываются исходя из следующих размеров выплат:

– пенсионный фонд – 22%;

– фонд социального страхования - 2,9%;

– федеральный фонд обязательного медицинского страхования – 5,1 %

Итого- 30%.

Оплата труда и социальные отчисления. В соответствии с ранее приведенными этапами оплата группы проекта составит:

– руководитель проекта. Количество дней занятости в проекте составляет 36 дней. Заработная плата одного дня при средней ежемесячной заработной плате в 38000 рублей - – 1809,52 рубля. Следовательно, оплата труда составит 65142,72

рубля и отчисления в социальные фонды (30%) 19542,816 рублей, всего 84685,536 рублей;

– студент (Программист). Количество занятости дней в проекте 102. Оплата одного дня работы в проекте при средней ежемесячной заработной плате в 25000 рублей – 1190,47 рублей. Следовательно, оплата труда программиста составит 121428,57 рублей. Социальные отчисления составят 36428,571 рублей, всего 157857,141 рублей. Стоимость программного обеспечения для разработки – php – не учитывается, так как оно бесплатно.

Совокупные затраты на оплату труда составят:

С =84685,536+157857,141=242542,677 рублей.

2. Накладные расходы и расходы на электроэнергию.

– накладные расходы: бумага, компакт диски, пользование Интернетом, картридж для принтера. В совокупности составляет 1500 рублей.

– затраты на электроэнергию составляют: 100 дней по 8 часов три компьютера программиста с затратой 0.3 киловатт/час по стоимости 5,38×100×8×0,3×2,35= 3034,32 рубля.

Совокупные накладные расходы и расходы на электроэнергию составляют: 4534,32 рубля.

Затраты на материалы приведены в Таблице 3.4.

Таблица 3.4 – Затраты на материалы

Наименование материала	Единицы измерения	Количество, шт.	Цена, руб.	Общая стоимость, руб.
Бумага	Пачка	10	100	1000
Картриджи для принтера	шт.	5	350	1750
Ручки шариковые	шт.	5	15	75
Карандаши с ластиком	шт.	12	15	180
Флэш-накопители	шт.	4	250	1000
CD-RW80 Verbatim	шт.	10	15	150

Итого:				4155

При разработке системы необходимо учесть амортизацию компьютера, которая составит 20% от его первоначальной стоимости (16000 рублей) при сроке эксплуатации 5 лет, то есть в сутки 3200/365= 8,76 рублей. Следовательно, для руководителя амортизация составит 36×8,76= 315,36 рублей, для программиста - 893,52 рублей, а всего - 1208,52 рублей.

Из выше представленных расчетов следует, что капитальные вложения на проектирование, разработку и внедрение составляют:

Зд=1208+4155+4534,32+242542,677=252439,997 рублей.

Диаграмма, характеризующая соотношение затрат, приведена на Рисунке 3.4.

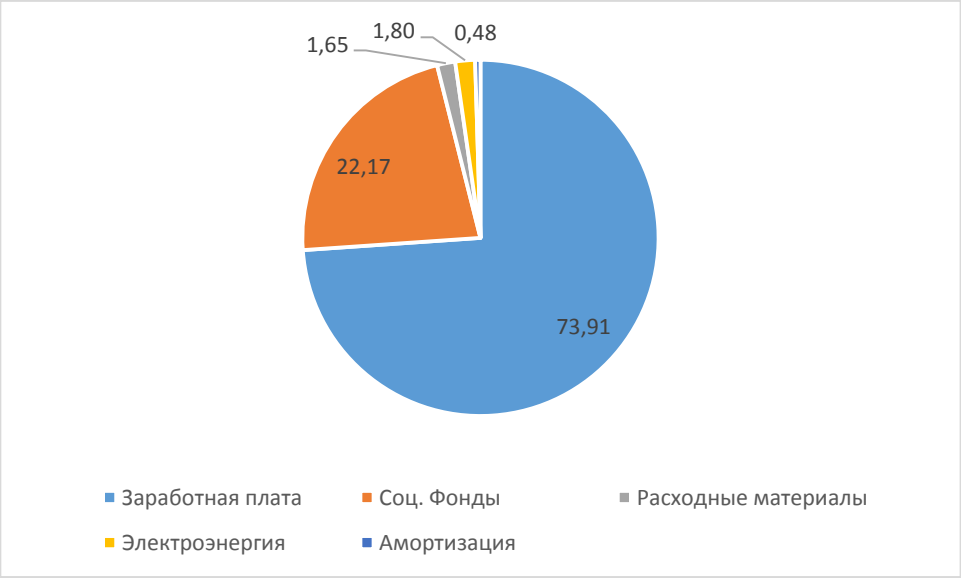

Рисунок 3.4 – Диаграмма, характеризующая соотношение затрат проекта (в процентах)

Для расчета прямого эффекта, опишем параметры выходных данных и стоимость затрат.

3.5 Расчет доходов и финансовых результатов

К трудовым показателям можно отнести следующие:

– абсолютное уменьшение трудовых затрат (ΔТ) (час), которое рассчитывается по формуле:

$$\Delta T = T_0 - T_{1,} \quad (1)$$

где Т0 – базовый вариант трудозатрат на обработку данных (час),

Т1 – расчет трудозатрат на обработку данных по предлагаемому варианту (час);

– параметр относительного уменьшения трудозатрат (Кт) (%), для подсчета которого используется формула:

$$К_Т = \Delta T / T_0 \times 100\%; \quad (2)$$

– индекс повышения производительности или уменьшения трудозатрат (YT), который можно высчитать по формуле:

$$I_T = T_0 / T \quad (3)$$

К стоимостным параметрам обычно относят:

– абсолютное уменьшение стоимости затрат (ΔС) (руб.), выводимое по формуле:

$$\Delta C = C_0 - C_1, \quad (4)$$

где С0 равно стоимостным затратам (в рублях) по базовому варианту обработку данных,

С1 выражает стоимостные затраты (в рублях) на реализацию обработки данных по предлагаемому варианту;

– коэффициент относительного уменьшения стоимостных расходов (КС) (%), выражающийся формулой:

$$К_С = \Delta С / С_0 \times 100\%; \quad (5)$$

– индекс повышения производительности или уменьшения трудозатрат (Yc), который можно высчитать по формуле:

$$I_С = С_0 / С \quad (6)$$

Подсчет вышеперечисленных показателей позволяет предположить вычисление частных показателей, например:

– трудоемкость отдельно взятой i-ой операции (Ti):

$$Ti = Vi / Ni, \quad (7)$$

где V_i выражает объем выполняемых на i-ой операции работ (действий или символов),

N_i равняется норме выполнения объема работ на каждой i-ой операции в один час;

– стоимостные затраты на реализацию каждой i-ой операции (C_i):

$$C_i = C_{з/п} + C_{нр} = T_i \times P_i \times (1+ К_{нр}), \quad (8)$$

где $C_{з/п}$ является стоимостными расходами на заработную плату сотруднику, выполнившему каждую i-ую операцию (руб.);

$C_{нр}$ выражает сумму накладных расходов (руб.);

P_i рассчитывает часовую тарифную ставку сотрудника, реализующего i-ую операцию (руб.);

$К_{нр}$ выражает коэффициент финансовых потерь, возникающих при выполнении i-ой операции;

– суммарная стоимость машинной работы по времени:

$$С_{машвр} = \sum_{m=1}^{M} (C_m^{маш.вр} * t_m^{раб.маш}),$$

(9)

где $C_m^{маш.вр}$ является стоимостью работы за каждую единицу времени на m-ом компьютере;

$t_m^{раб.маш}$ выражает общую продолжительность работы m-того компьютера.

Для расчета трудозатрат на обработку данных по базовому варианту ($Т_0$) (час) используют следующую формулу:

$$T_o = \sum T_{0i},$$

(10)

где $Т_{0i}$ определяет трудоемкость реализации i-ой операции при базовом варианте решения задачи.

Для подсчета трудозатрат на обработку данных по предлагаемому варианту ($Т_1$) (час) нужно пользоваться следующим алгоритмом:

$$T_1 = \sum T_{1j},$$

(11)

где $Т_{1j}$ является трудоемкостью исполнения j-ой операции для предлагаемого варианта решения.

Чтобы просчитать стоимостные затраты на обработку данных по проектному варианту ($С_1$) (руб.), используется формула:

$$C_1 = \sum C_{1j} + \quad С_1 \quad _{машвр,} \quad + \quad К_{экпс,} \quad (12)$$

где $С_{1j}$ отражает стоимостные затраты на реализацию j-ой операции для предлагаемого варианта решения.

К главным обобщающим показателям экономической эффективности можно отнести: годовой экономический эффект, расчетный параметр эффективности капитальных вложений, время окупаемости затрат на запуск рассматриваемого проекта.

Годовой экономический эффект от создания и запуска информационной системы ($Э_г$) определяется разностью абсолютного снижений стоимостных затрат и предполагаемой прибылью:

$$Э_г = \Delta С – К_П * Е_н , \quad (13)$$

где $К_П$ отражает расходы на реализацию проекта автоматизированной обработки данных, руб.;

$Е_н$ является прописанным параметром эффективности капитальных вложений.

В данном случае произведение $К_П$ и $Е_н$ можно рассматривать как норму прибыли, которая получается при внедрении системы. Параметр $Е_н$ обычно равен 0,15 для любой отрасли народного хозяйства. Ен также можно рассматривать как минимальную эффективность капитальных вложений, ниже которой они уже нерентабельны.

Высчитанное значение показателя годового экономического эффекта ($Э_г$) в данном случае способствует сопоставлению экономических итогов автоматизации

обработки информации с результативностью капитальных вложений в остальные направления улучшения производства и управления им.

Расчетный параметр эффективности капитальных затрат ($E_р$) обозначает отношение абсолютного снижения затрат стоимостных к затратам на реализацию и проектирование системы:

$$E_р = \Delta C / К_П. \quad (14)$$

Время окупаемости ($Т_{ОК}$) обычно является неким отношением капитальных затрат на создание и установку ИС к максимальному снижению стоимостных затрат:

$$Т_{ОК} = К_П / \Delta C. \quad (15)$$

Срок окупаемости затрат на внедрение проекта машинной обработки информации:

$Т_{ок}$ = 252439/61996=4 года

Окупаемость затрат на внедрение проекта составляет около четырех лет.

Выводы по главе три:

Для ИП выгоднее разработать собственную конфигурацию, которую они могут сопровождать самостоятельно, так как при приобретении схожих систем, есть возможность получить излишний функционал, и их покупка не будет оправдана. Внедрение данной информационной системы для ИП окупаемо и экономически выгодно, так как оно окупится в течение четырех лет.

ЗАКЛЮЧЕНИЕ

Цель данной выпускной квалификационной работы – разработка информационной системы учета телефонных разговоров сотрудников.

Исходя из цели и поставленных задач, была определена структура данной выпускной квалификационной работы, в ходе которой был произведён анализ деятельности ИП, анализ алгоритмов для создания информационной системы, проведен анализ инструментария, представлена программная реализация для написания и описан результат работы программного продукта.

Результатом решения поставленных задач является создание информационной системы по учету компьютерного оборудования, который способен сократить временные и человеческие ресурсы.

В начале выполнения выпускной квалификационной работы были изложены теоретические основы построения систем учета.

Рассмотрены известные аналогичные разработки, кратко описана их функциональность и приведены иллюстрации их интерфейса. Система-прототип описана со всеми видами обеспечения: организационное, информационное, техническое и программно- алгоритмическое.

Таким образом поставленная задача выполнена в полном объеме – цель выпускной квалификационной работы достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Аникеев, С.В. Разработка приложений баз данных в Delphi / С.В.Аникеев, А.В.Маркин. – Москва : Изд-во Диалог-МИФИ, 2013. – 160 с.
2. Виллемер, А. Программирование на C++ / Арнольд Виллемер; пер. с анг. М. Райтмана – Москва: Изд-во Эксмо, 2013 г. – 528 с.
3. Бондарев, В.В. Введение в информационную безопасность автоматизированных систем: учебное пособие / В.В. Бондарев. – Москва: Изд-во МГТУ им. Н. Э. Баумана, 2018. – 252 с.
4. Тейт, Б. Семь языков за семь недель. Практическое руководство по изучению языков программирования / Брюс Тейт – Москва: Изд-во ДМК-Пресс, 2014. – 384 с.
5. Васвани, В.А. MySQL: использование и администрирование / В.А. Васвани. – СПб: Изд-во Питер, 2011. – 368 с.
6. Васильев А.В. Самоучитель C++ с примерами и задачами / А.В. Васильев – Москва: Изд-во Наука и Техника, 2015. – 480 с.
7. Методические рекомендации по подготовке и оформлению выпускной квалификационной работы (проекта) для технических направлений подготовки 09.03.01 Информатика и вычислительная техника, 09.03.04 Программная инженерия, 12.03.01 Приборостроение, 23.03.01 Технология транспортных процессов / сост. Л.Н.Буйлушкина. - Нижневартовск, 2017. – 35 с.
8. Дейтел, П. Как программировать на Visual C# 2012. Включая работу на Windows 7 и Windows 8 / Пол Дейтел, Харви Дейтел – Москва: Изд-во Питер, 2014. – 864 с.
9. Майо, Д. Самоучитель MicrosoftVisualStudio 2011 / Джо Майо – Москва: Изд-во BHV, 2011. – 464 с.
10. Залогова, Л. А. Основы объектно-ориентированного программирования на базе языка C# / Л.А. Залогова – Москва: Изд-во Лань, 2018. – 192 с.

11. Зоткин, С. А. Программирование на языке высокого уровня С/С++. Конспект лекций / С.А. Зоткин – Москва: Изд-во МГСУ, 2018. – 140 с.

12. Исаев, Г.А. Информационные системы в экономике. Учебник / Г.А. Исаев – Москва: Изд-во Омега-Л, 2013. – 462 с.

13. Исаев, Г.А. Проектирование информационных систем. Учебное пособие / Г.А. Исаев – Москва: Изд-во Омега-Л, 2015. – 424 с.

14. Гриффитс, И. Программирование на C# 5.0 / Иэн Гриффитс – Москва: Изд-во Эксмо, 2014. – 1135 с.

15. Кузьменко, Е.А. Информатика. Углубленный курс. Учебное пособие для СПО / Е.А Кузьменко, О.А. Мойзес – Москва: Изд-во Юрайт, 2018. – 164 с.

16. Курлов, А.А. Методология ин формационной аналитики / А.А Курлов, Е.А. Петров – Москва: Изд-во Проспект, 2014. – 384 с.

17. Мартынов, Н.А. Программирование для Windows на C\C++. В 2-х томах / Н.А. Мартынов – Москва: Изд-во Бином, 2013. – 480 с.

18. Никифоров, С.Н. Прикладное программирование / С. Н. Никифоров – Москва: Изд-во Лань, 2018. – 140 с.

19. Ошероув, Р.О. Искусство автономного тестирования с примерами на C# / Р.О. Ошеуров – Москва: Изд-во ДМК-Пресс, 2014. – 360 с.

20. Паттерсон, Д. Архитектура компьютера и проектирование компьютерных систем / Дэвид Паттерсон, Джон Хеннесси – Москва: Изд-во Классика ComputersScience, 2012. – 784 с.

21. Полубенцева, М.П. С/ С++ Процедурное программирование / М.П. Полубенцева – Москва: Изд-во BHV, 2014. – 432 с.

22. Потопахин В.С. Искусство алгоритмизации / В.С. Потопахин – Москва: Изд-во ДМК-Пресс, 2014. – 320 с.

23. Пугачев, Е.К. Разработка приложений для Windows 8 на языке C# / Е.К. Пугачев, Ш.К. Шериев, Е.А. Кичинский – Москва: Изд-во BHV, 2013. – 416 с.

24. Мюллер, Р. Дж. Проектирование баз данных и UML / Роберт Дж. Мюллер – Москва: Изд-во Лори, 2013. – 432 с.

25. Лафоре, Р. Объектно-ориентированное программирование в C++ / Роберт Лафоре – Москва: Изд-во Питер, 2013. – 928 с.

26. Сурядный, А.Е. Microsoft Access 2011. Лучший самоучитель / А.Е Сурядный – Москва: Изд-во Астрель, 2012. – 448 с.

27. Таненбаум, А.А. Современные операционные системы / А.А. Таненбаум, Е.В. Бос – Спб: Изд-во Питер, 2015. – 1120 с.

28. Стиллмен, Э. Изучаем C# / Эндрю Стиллмен – Москва: Изд-во Питер, 2014. – 816 с.

29. Тимофеев, В. А. Самоучитель C++ как он есть / В.А. Тимофеев – Москва: Изд-во Бином, 2009. – 336 с.

30. Тюгашев, А. Е. Языки программирования. Учебное пособие. Стандарт третьего поколения / А.Е. Тюгашев – Москва: Изд-во Питер, 2014. – 336 стр.

31. Филимонова, Е.В. Информатика и информационные технологии в профессиональной деятельности. Учебник / Е.В. Филимонова – Москва: Изд-во Юстиция, 2019. – 216 с.

32. Ховард, М. Как написать безопасный код на C++, Java, Perl, PHP, ASP.NET / Майкл Ховард, Дэвид Лебланк, Джон Виега – Москва: Изд-во ДМК-Пресс, 2014. – 288 с.

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ А. ФРАГМЕНТ ЛИСТИНГА ПРОГРАММНЫХ МОДУЛЕЙ

```
<?php

namespace app\controllers;

use Yii;
use app\models\PhoneNumber;
use app\models\PhoneNumberSearch;
use yii\web\Controller;
use yii\web\NotFoundHttpException;
use yii\filters\VerbFilter;
use yii\filters\AccessControl;

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

```
                    'rules' =>
                    [
```

Продолжение приложения А

```
                        [
                            'actions'    =>    ['index',    'view',
'create', 'update', 'delete'],
                            'allow' => true,
                            'roles' => ['@'],

                            'matchCallback'   =>   function   ($rule,
$action)
                            {
                                return   Yii::$app->user->identity-
>IsAdmin;
                            }

                        ],
                    ],
                ],

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


        /**
         * Действие "index" генерирует страницу вывода списка всех
моделей класса "PhoneNumber".
```

```
         * @return mixed
         */
```



```
        public function actionIndex()
        {
            $searchModel = new PhoneNumberSearch();
            $dataProvider = $searchModel->search(Yii::$app->request->queryParams);

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

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

        /**
         * Действие "create" создает новый экземпляр класса "PhoneNumber" (новую модель).
```

```
         * Если операция успешно выполнена, браузер будет
перенаправлен на страницу просмотра (действие 'view').
```



```
         * @return mixed
         */
        public function actionCreate()
        {
            $model = new PhoneNumber();

            if ($model->load(Yii::$app->request->post()) && $model-
>save()) {


                return $this->redirect(['view', 'id' => $model-
>id]);
            }
          $model->registration_date = date('Y-m-d H:i:s');
            return $this->render('create', [
                'model' => $model,
            ]);
        }

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

```
            $model = $this->findModel($id);
```

Продолжение приложения А

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

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

          $model->delete();

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

        /**
```

```
         * Метод поиска модели класса "PhoneNumber" по первичному
ключу.
         * Если модель не найдена, выдает исключение HTTP 404.
         * @param integer $id
```



```
         * @return Объект класса "PhoneNumber" (модель)
         *      @throws      Возвращает      исключение      класса
"NotFoundHttpException", если модель не найдена
         */
        public static function findModel($id)
        {
          $model = PhoneNumber::findOne($id);
            if ($model !== null) {
                return $model;
            }

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

    namespace app\controllers;

    use Yii;
    use app\models\Connection;
    use app\models\ConnectionSearch;
    use yii\web\Controller;
    use yii\web\NotFoundHttpException;
    use yii\filters\VerbFilter;
    use yii\filters\AccessControl;

    /**
```

```
     * Класс "ConnectionController" реализует действия CRUD (сокр. от англ. create, read, update, delete — «создать, прочесть, обновить, удалить») для модели класса "Connection".
     */
    class ConnectionController extends Controller
```

Продолжение приложения А

```
    {
        /**
         * Метод, определяющий поведение контроллера (правила доступа к действиям контроллера)
         */
     public function behaviors()
        {
            return [
               'access' => [
                    'class' => AccessControl::className(),
                    'rules' =>
                    [
                         [
                              'actions'    =>    ['index',    'view', 'create', 'update', 'delete'],
                              'allow' => true,
                              'roles' => ['@'],

                              'matchCallback'   =>   function   ($rule, $action)
                              {
                                   return   Yii::$app->user->identity->IsAdmin;
                              }

                         ],
                    ],
               ],
```

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



```
                    ],
                ],
            ];
        }


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

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

        /**
         * Действие "view" отображает данные конкретной модели класса "Connection".
         * @param integer $id
         * @return mixed
```

```
         *      @throws      Возвращает      исключение      класса
"NotFoundHttpException", если модель не найдена
         */
        public function actionView($id)
        {
```

Продолжение приложения А

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

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

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


          $model->registration_date = date('Y-m-d H:i:s');
          $model->begin_date     =          date('Y-m-d     H:i:s',
strtotime($model->registration_date  .  '  +'  .  rand(1,  15)  .  '
second'));
```

```
            $model->end_date = date('Y-m-d H:i:s', strtotime($model->begin_date . ' +' . rand(10,500) . ' second'));

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

Продолжение приложения А

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

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

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

        /**
```

```
         * Действие "delete" - удаление выбранной модели класса "Connection".
         * Если операция успешно выполнена, браузер будет перенаправлен на страницу 'index'.
         * @param integer $id
         * @return mixed
         *      @throws      Возвращает      исключение      класса "NotFoundHttpException", если модель не найдена
```

Окончание приложения А

```
         */
        public function actionDelete($id)
        {
            $model = $this->findModel($id);

          $model->delete();

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

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

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

ПРИЛОЖЕНИЕ Б. КОМПАКТ ДИСК

Содержание:

1. ПЗ ВКР Дмитриева Б. В. НвФл-422.
2. Презентация.
3. Файлы программы.