Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра информатики

ОТЧЕТ

по преддипломной практике

ОГУ 09.03.02.7223.223 П

Руководитель от кафедры
канд. пед. наук, доцент _____ Т. Е. Тлегенова

Руководитель от предприятия
руководитель группы
системных инженеров г.
Оренбурга «Уральский центр
систем безопасности» _____ А. В. Манжосов

Студент группы
19ИСТ(б)ОП _____ В. К. Чернышов

Оренбург 2023

Федеральное государственное бюджетное образовательное учреждение высшего образования «Оренбургский государственный университет» (ОГУ)

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ НА ПРАКТИКУ

Вид, тип практики преддипломная
Обучающийся Чернышов Владимир Константинович
(Фамилия, Имя, Отчество)

Курс 4
Факультет (филиал, институт) Институт математики и информационных технологий
Форма обучения очная
Направление подготовки 09.03.02. – Информационные системы и технологии
Содержание задания на практику (перечень подлежащих рассмотрению вопросов, выполняемых работ, связанных с будущей профессиональной деятельностью):
1 Описать профильную организацию ООО «УЦСБ»;
2 Проанализировать предметную область;
3 Построить модели информационных процессов;
4 Обосновать выбор инструментальных средств проектирования и разработки информационной системы;
5 Спроектировать базу данных;
6 Разработать приложение;
7 Разработать документацию по использованию информационной системы;
8 Оформить отчет по практике.

Дата выдачи задания 07.04.2023
Руководитель практики от Университета _____ Т. Е. Тлегенова
подпись И.О. Фамилия

Согласовано:

Руководитель практики от
Профильной организации _____ А. В. Манжосов
подпись И.О. Фамилия

Ознакомлен:

Обучающийся _____ В. К. Чернышов
подпись И.О. Фамилия

Заключение руководителя о выполнении задания практики:

_____

_____

Руководитель практики от Университета _____ Т. Е. Тлегенова
подпись И.О. Фамилия

Рабочий график (план) проведения практики

Вид, тип практики преддипломная

Обучающийся Чернышов Владимир Константинович
(Фамилия, Имя, Отчество)

Курс 4
Факультет (филиал, институт) Институт математики и информационных технологий
Форма обучения очная
Направление подготовки (специальность) 09.03.02 Информационные системы и технологии
Место прохождения практики ООО «Уральский центр систем безопасности»
(наименование профильной организации)

Срок прохождения практики: с 10.04.2023 по 20.05.2023
Руководитель практики от ОГУ Т.Е. Тлегенова, канд. пед. наук
Руководитель практики от организации А.В. Манжосов,
Руководитель группы системных инженеров г. Оренбурга «Уральский центр систем безопасности»

Дата (период)	Содержание и планируемые результаты практики
10.04	Вводный инструктаж по технике безопасности на рабочем месте
	Ознакомление с историей развития, структурой и задачами предприятия ООО «УЦСБ»
11.04-14.04	Анализ предметной области
17.04-21.04	Исследование моделей информационных процессов информационной системы
24.04 - 12.05	Проектирование базы данных
	Разработка приложения
15.05 - 18.05	Завершение разработки документации, разработки руководства
19.05	Оформление отчета, разработка презентации

Руководитель практики от Университета _____ Т. Е. Тлегенова
подпись И.О. Фамилия

Руководитель практики от
Профильной организации _____ А. В. Манжосов
подпись И.О. Фамилия

Аннотация

В данном отчете описан процесс прохождения преддипломной практики на предприятии ООО «УЦСБ». Основным видом деятельности является создание, модернизация и обслуживание подсистем защиты, а также других услугах в сфере информационной безопасности в организациях и предприятиях.

Отчет состоит из 4 разделов.

В первом разделе производится описание структуры предприятия, его схемы сети и требуемой к реализации цели.

Во втором разделе проводится анализ предметной области. Указываются сведения по теме индивидуального задания, определяется используемый метод решения.

В третьем разделе реализуется проектирование и разработка информационной системы.

В четвертом разделе реализуется разработка руководства пользователя и администратора системы, а также проводится тестирование информационной системы.

Отчет содержит 75 страниц, 54 рисунка.

Содержание

Введение.....6
1 Общая характеристика профильной организации.....7
1.1 Организационная структура предприятия.....7
1.2 Основные задачи отделов предприятия.....8
1.3 Требование на индивидуальное задание.....9
2 Анализ предметной области.....10
2.1 Актуальность СППР.....10
2.2 Актуальность замены сетевого оборудования.....12
2.3 Классификация методов принятия решения.....13
2.4 Описание используемого метода.....16
2.5 Анализ информационных потоков разрабатываемой информационной системы.....19
2.6 Объектно-ориентированное моделирование информационной системы....25
3 Разработка информационной системы.....35
3.1 Выбор СУБД.....35
3.2 Проектирование базы данных.....36
3.3 Выбор среды разработки.....40
3.4 Разработка приложения.....41
4 Разработка документации по использованию информационной системы.....48
4.1 Руководство для пользователя.....48
4.2 Руководство для администратора системы.....51
4.3 Тестирование разработанного приложения.....55
Заключение.....58
Список использованных источников.....60
Приложение А (обязательное).....62

Введение

Преддипломная практика студентов образовательных учреждений является завершающей стадией учебного процесса, помогающая подготовить будущего специалиста к профессиональной деятельности, проводится на промышленных предприятиях, в организациях и фирмах. Продолжительность практики – 6 недель.

Основная цель преддипломной практики – закрепление и углубление теоретических знаний, полученных студентами в процессе обучения, а также углубленное исследование темы диплома.

Эта цель достигается в результате знакомства с работой предприятия, приобретением навыков профессиональной и организационной деятельности на рабочих местах, участия в решении практических проблем.

На основании поставленной цели было выделено несколько задач:

1 Описать профильную организацию ООО «УЦСБ»;

2 Проанализировать предметную область;

3 Построить модели информационных процессов;

4 Обосновать выбор инструментальных средств проектирования и разработки информационной системы;

5 Спроектировать базу данных;

6 Разработать приложение;

7 Разработать документацию по использованию информационной системы;

8 Оформить отчет по практике.

1 Общая характеристика профильной организации

1.1 Организационная структура предприятия

ООО «УЦСБ» – Уральский центр систем безопасности – организация, зарегистрированная 8 мая 2007 года [1] и специализирующаяся на создании, модернизации и обслуживании подсистем защиты, а также других услугах в сфере информационной безопасности в организациях и предприятиях [2].

Организационная структура предприятия – это документ, в котором схематически отражается состав и иерархия внутренних подразделений организации [3]. Благодаря данному документу мы можем получить представление о самом предприятии. Также, данный документ как бы описывает работу организации изнутри, благодаря чему мы можем проследить взаимосвязи между сотрудниками, отделами, направлениями и руководством, сферы компетенции, а также их права и обязанности.

Рассмотрим организационную структуру ООО «УЦСБ», представленную на рисунке 1.1.

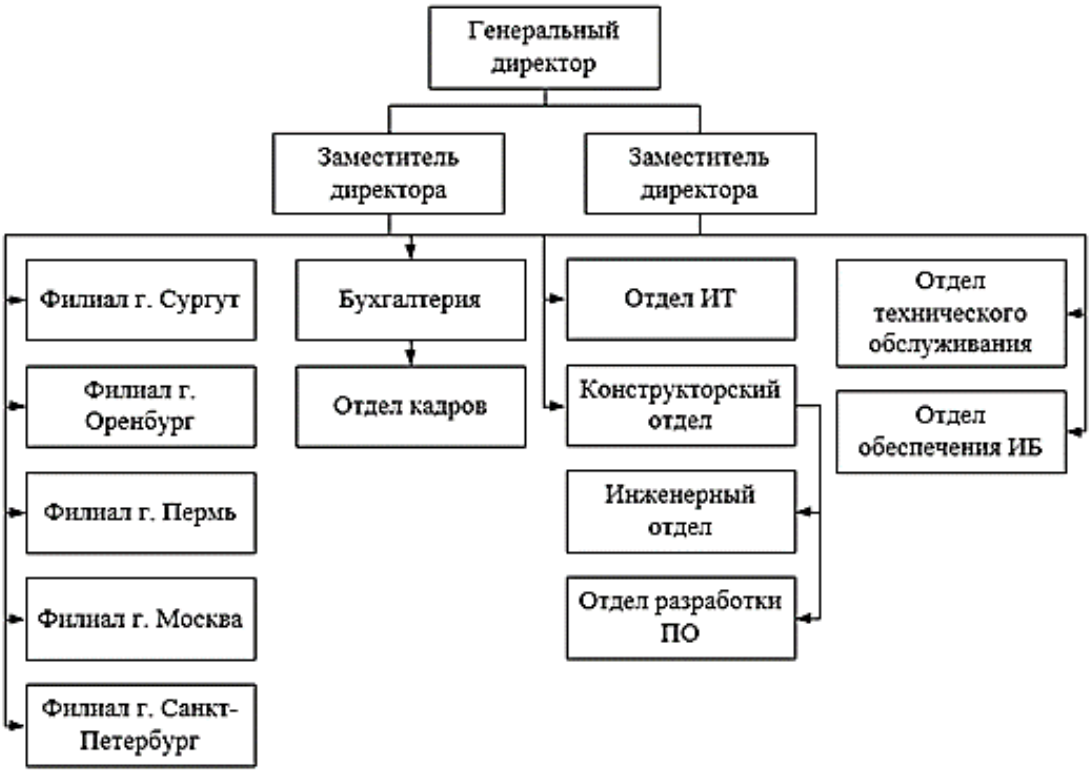

Рисунок 1.1 – Организационная структура ООО «УЦСБ»

1.2 Основные задачи отделов предприятия

На основании разработанной организационной структуры определим обязанности каждого отдела. Руководителем является генеральный директор, который имеет самые высокие привилегии и обеспечивает выполнение возложенных на него задач, организует работу и эффективное взаимодействие всех структурных подразделений. Должность заместителя директора сама по себе является очень важной, ведь это фактически второй по значимости человек в организации, а в случае отсутствия директора все решения принимаются одним из заместителей директора.

Приведем краткое описание каждого из отделов:

1 Бухгалтерия. Данный отдел предназначен для учета финансового и хозяйственного имущества. Благодаря деятельности данного отдела в организации имеются документальные, структурированные обоснования необходимые для дальнейшей экономической деятельности. Другими словами, именно основываясь на результатах деятельности этого отдела могут приниматься решения о продлении или расторжении контрактов, начала или окончания разработки какого-либо продукта и т.д.

2 Отдел кадров способствует достижению целей организации путем обеспечения организации необходимыми кадровыми единицами. Благодаря деятельности отдела кадров организация в кратчайшие сроки может найти необходимый в данный момент персонал в кратчайшие сроки.

3 Основная задача отдела информационных технологий (отдела ИТ) – это обеспечить бесперебойную деятельность оборудования, благодаря которому организация может выполнять свои основные функции. Помимо своей основной деятельности данный отдел также занимается оснащением информационной безопасности внутренних процессов.

4 Конструкторский отдел подразделяется на два подраздела, а именно: инженерный отдел, инженеры которого разрабатывают аппаратное обеспечение основных продуктов, выпуском которых занимается организация (например, программно-аппаратный комплекс обеспечения безопасности АСУ ТП DATAPK), и отдел разработки программного обеспечения (ПО), задачами которого является качественное и своевременное обеспечение программным обеспечением ранее разработанных аппаратных продуктов, в случае, если это необходимо.

5 Отдел технического обслуживания обеспечивает обслуживание уже существующих клиентов организации в случае, если возникает необходимость в настройке оборудования или обратной связи.

6 Отдел обеспечения информационной безопасности занимается вопросами обеспечения информационной безопасности инженерно-технической инфраструктуры, системам передачи данных и вычислительных системам, систем обработки и хранения данных, а также комплексными системами информационной безопасности.

Так же в состав организации входят филиалы, расположенные в таких городах, как Сургут, Оренбург, Пермь, Москва и Санкт-Петербург, чья

организационная структура схожа со структурой головного отделения, за исключением конструкторского отдела.

На рисунке 1.2 представлена схема сети оренбургского филиала организации.

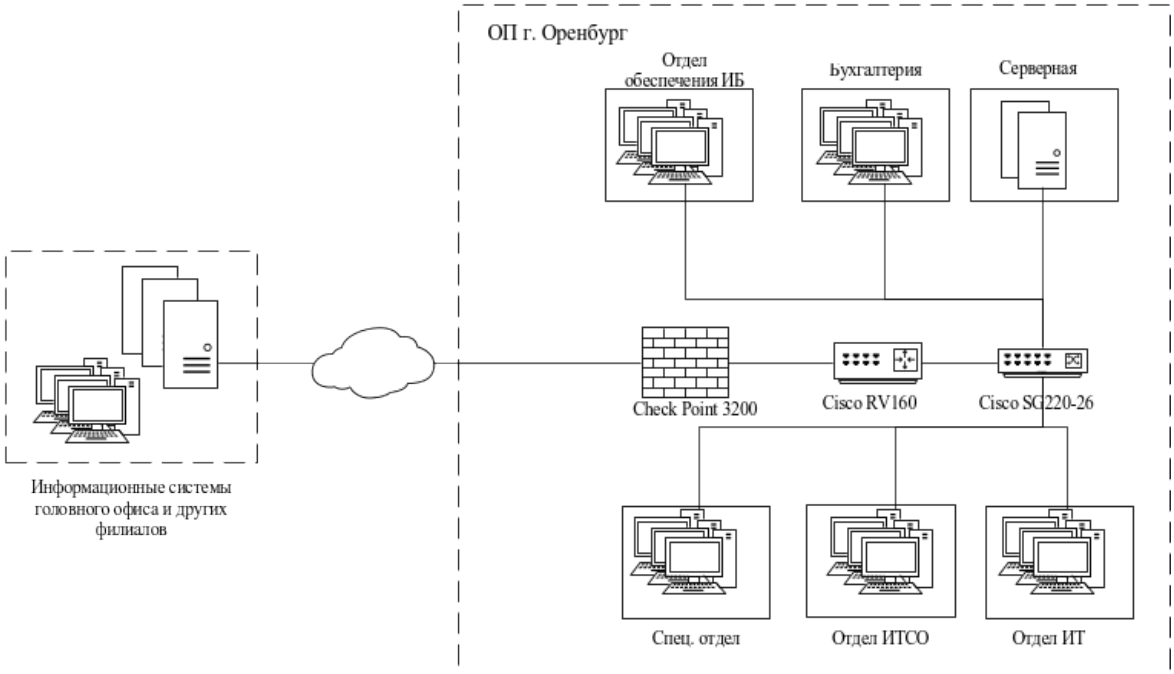

Рисунок 1.2 – Схема сети ООО «УЦСБ» г. Оренбург

1.3 Требование на индивидуальное задание

В качестве индивидуального задания было принято участие в разработке информационной системы поддержки принятия решений по замене сетевого оборудования организации.

2 Анализ предметной области

2.1 Актуальность СППР

Современная картина российского рынка характеризуется всевозрастающим беспорядком и сомнительной конкуренцией различного рода продукции. Стремительное развитие такого сектора как информационные технологии стимулирует ускоренный рост темпа происходящих изменений в современных компаниях. Поэтому основными факторами, обеспечивающими конкурентное преимущество, являются скорость анализа большего количества информации, быстрота принятия решений и точность итогового выбора.

Система поддержки принятия решений (Далее – СППР) предполагает достаточно глубокую проработку данных, специально преобразованных так, чтобы их было удобно использовать в ходе процесса принятия решений. Неотъемлемым компонентом СППР являются правила принятия решений, которые на основе агрегированных данных подсказывают управленческому составу выводы.

Качество СППР в первую очередь зависит от данных, на основании которых принимаются решения; используемых аналитических методов и моделей обработки и анализа данных; адекватности используемых инструментальных средств задачам принятия решений [4].

Система поддержки принятия решений предназначена для поддержки многокритериальных решений в сложной информационной среде. При этом под многокритериальностью понимается тот факт, что результаты принимаемых решений оцениваются не по одному, а по совокупности многих показателей (критериев) рассматриваемых одновременно. Информационная сложность определяется необходимостью учета большого объема данных, обработка которых без помощи современной вычислительной техники практически невыполнима. В этих условиях число возможных решений, как правило, весьма велико, и выбор наилучшего из них "на глаз", без всестороннего анализа может приводить к грубым ошибкам.

Система поддержки принятия решений решает две основные задачи:

1 Выбор наилучшего решения из множества возможных (оптимизация).

2 Упорядочение возможных решений по предпочтительности (ранжирование).

Процесс принятия решения – это преобразование исходной информации в выходную информацию [5].

Процесс принятия решений является сложной итерационной процедурой. Структурная схема процесса принятия решений изображена на рисунке 2.1.

Решение проблем – процесс, ибо речь идет о нескончаемой последовательности взаимосвязанных шагов. Руководитель заботится не столько о решении как таковом, сколько обо всем, связанным и проистекающем из него. Для решения проблемы требуется не единичное решение, а совокупность выборов.

Рисунок 2.1 – Структурная схема процесса принятия решения

1 Диагностика проблемы. Первый шаг на пути решения проблемы – ее определение. Существуют два способа рассмотрения проблемы. Согласно одному, проблемой считается ситуация, когда поставленные цели не достигнуты. Как проблему можно рассматривать также потенциальную возможность. Например, поиск для замены устаревшего сетевого оборудования.

2 Формулировка ограничений и критериев принятия решения. Многие возможные решения проблем организации не будут реалистичными, поскольку либо у руководителя, либо у организации недостаточно ресурсов для реализации принятых решений. Ограничения корректирующих действий сужают возможности в принятии решений.

В дополнении к идентификации ограничений, необходимо определить стандарты, по которым предстоит оценивать альтернативные варианты выбора. Эти стандарты принято называть критериями принятия решений. Они выступают в качестве рекомендаций по оценке решений.

3 Определение альтернатив. Следующий этап – формулирование набора альтернативных решений проблемы.

4 Оценка альтернатив. При их выявлении необходима определенная предварительная оценка. При оценке решений определяются достоинства и недостатки каждого из них, и возможные общие последствия.

5 Выбор альтернативы. Если проблема была правильно определена, а альтернативные решения тщательно взвешены и оценены, сделать выбор, т. е. принять решение сравнительно просто.

6 Реализация. Процесс решения проблемы не заканчивается выбором альтернативы. Простой выбор направления действий имеет малую ценность для организации. Для разрешения проблемы или для извлечения выгоды из имеющейся возможности решения должно быть реализовано. Уровень эффективности осуществления решения повысится, если оно будет признано теми, кого оно затрагивает.

7 Обратная связь. Еще одной фазой, входящей в процесс принятия решения и начинающейся после того, как решение начало действовать, является установление обратной связи. Обратная связь – т. е. поступление данных о том, что происходило до и после реализации решения – позволяет скорректировать решение, пока организации не нанесено значительного ущерба.

Таким образом, концепция СППР включает целый ряд средств, объединенных общей целью – способствовать принятию рациональных и эффективных управленческих решений.

СППР – компьютерная автоматизированная система, целью которой является помощь людям, принимающим решение в сложных условиях для полного и объективного анализа предметной деятельности. Это диалоговая система, использующая правила принятия решений и соответствующие модели с базами данных, а также интерактивный компьютерный процесс моделирования. СППР возникли в результате слияния управленческих информационных систем и систем управления базами данных. СППР являются человеко-машинными системами, которые позволяют лицам, принимающим решения, использовать данные, знания, объективные и субъективные модели для анализа и решения неструктурированных и слабо формализуемых задач. Процесс принятия решения – это получение и выбор наиболее оптимальной альтернативы с учетом просчета всех последствий. При выборе альтернатив – надо выбирать ту, которая наиболее полно отвечает поставленной цели, но при этом приходится учитывать большое количество противоречивых требований и, следовательно, оценивать выбранный вариант решения по многим критериям [10].

2.2 Актуальность замены сетевого оборудования

Модернизация или замена сетевого оборудования (Далее – СО) выполняется для закрытия различных целей предприятия. Наиболее частая из них заключается в увеличении пропускной способности и скорости вычислительной сети. Другая распространенная причина модернизации – появление у предприятия филиалов, которым необходимо предоставить доступ к базе файлов. Также СО модернизируют для организации удаленной работы и

осуществления доступа к документам предприятия для сотрудников, находящихся в командировке.

Специалисты рекомендуют обновлять оборудование на постоянной основе. Точные сроки зависят от компании, ее сферы деятельности, размера, перспектив роста и расширения локальной сети, количества используемых и планируемых к внедрению сетевых технологий, размера бюджета и сотни других факторов. Однако существует несколько критериев, которые помогут понять, насколько необходима замена сетевого оборудования, а также аргументировать это перед собственником.

Использование современных компьютерных технологий дает возможность поддерживать на высоком уровне техническое обеспечение рабочих мест предприятия. Модернизация СО позволяет решать проблемы программной и аппаратной части. Их возникновение одинаково вероятно и требует немедленного устранения, пока неполадки не привели к остановке работы предприятия. Наиболее часто возникающие проблемы, требующие модернизации:

- работа служб DHCP и DNS нарушена;
- настройка паролей и учетных записей сопровождается сложностями;
- настройки Firewall некорректны;
- программное обеспечение требует обновления;
- конфигурация коммутатора неверна;
- кабель физически поврежден;
- коннекторы слабо обжаты;
- сетевые карты сломаны;
- передача информации, а также формирование отчетов замедлены;
- почтовые службы, 1С, прочие сервисы работают медленно;
- связь между рабочими местами нарушена.

Наличие в компании нескольких из перечисленных признаков говорит о необходимости проведения модернизации компьютерной сети предприятия. Прежде, чем приступить к модернизации СО, проводят ее аудит для выявления причины возникновения уязвимостей и следствия возникших проблем.

В зависимости от того, какие будут выявлены недостатки, — аппаратные или программные, компании предложат несколько вариантов модернизации. Чаще всего модернизация происходит на базе внедренных ранее решений. От надежности функционирования вычислительной сети зависит работа всех бизнес-приложений, IP-телефонии или других программных продуктов.

2.3 Классификация методов принятия решения

Одним из главных предназначений информационных систем является сбор, обработка и предоставление информации для принятия решений.

В связи с этим методы обработки экономической информации удобно рассматривать по фазам жизненного цикла процесса принятия управленческого решения:

- диагностика проблем;

- разработка (генерирование) альтернатив;
- выбор решения;
- реализация решения [6].

Методы, используемые на фазе диагностики проблем, обеспечивают ее достоверное и наиболее полное описание. В их составе выделяют (рисунок 2.2) методы сравнения, факторного анализа, моделирования (экономико-математические методы, методы теории массового обслуживания, теории запасов, экономического анализа) и прогнозирования (качественные и количественные методы). Все эти методы осуществляют сбор, хранение, обработку и анализ информации, фиксацию важнейших событий. Набор методов зависит от характера и содержания проблемы, сроков и средств, которые выделяются на этапе постановки.

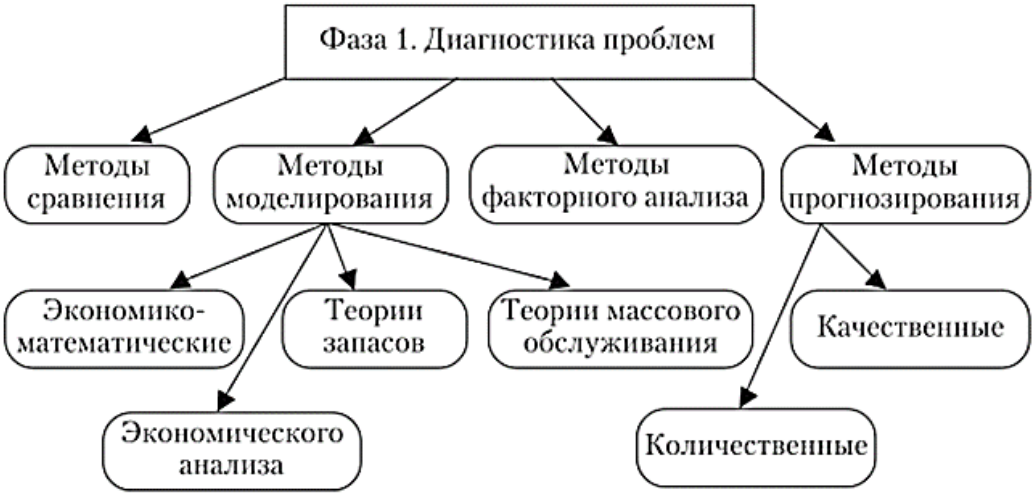

Рисунок 2.2 – Структурная схема фазы диагностики проблемы

Методы выявления (генерирования) альтернатив приведены на рисунке 2.3 На этой стадии также используются методы сбора информации, но в отличие от первого этапа, на котором осуществляется поиск ответов на вопросы тина «Что произошло?» и «По каким причинам?», здесь определяют, как можно решить проблему, с помощью каких управленческих действий.

При разработке альтернатив (способов управленческих действий по достижению поставленной цели) используют методы как индивидуального, так и коллективного решения проблем. Индивидуальные методы характеризуются наименьшими затратами времени, но не всегда эти решения являются оптимальными. При генерировании альтернатив используют интуитивный подход или методы логического (рационального) решения проблем.

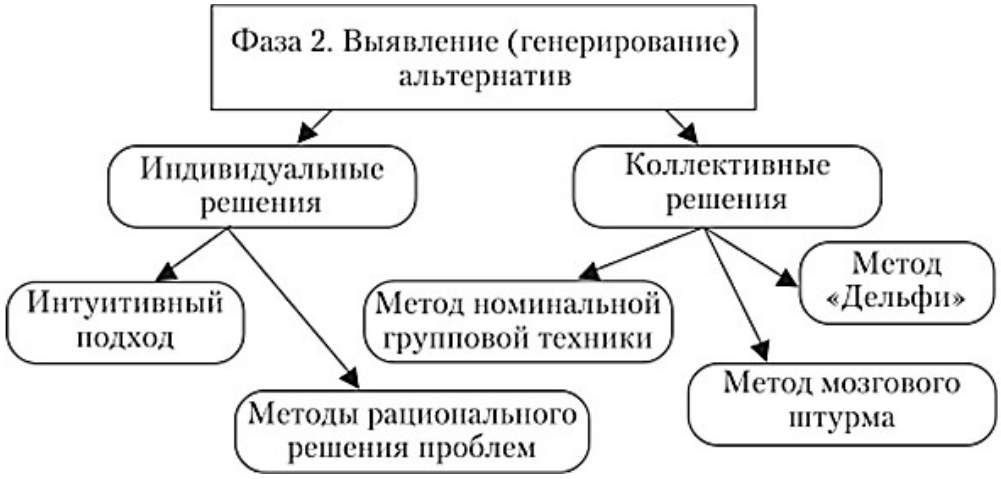

Рисунок 2.3 – Структурная схема фазы выявления альтернатив

Следующий этап, выбор альтернатив происходит чаще всего в условиях определенности, риска и неопределенности (рисунок 2.4). Отличие между этими состояниями среды определяется объемом информации, степенью знаний лицом, принимающим решение (Далее – ЛПР) сущности явлений, условий принятия решений.

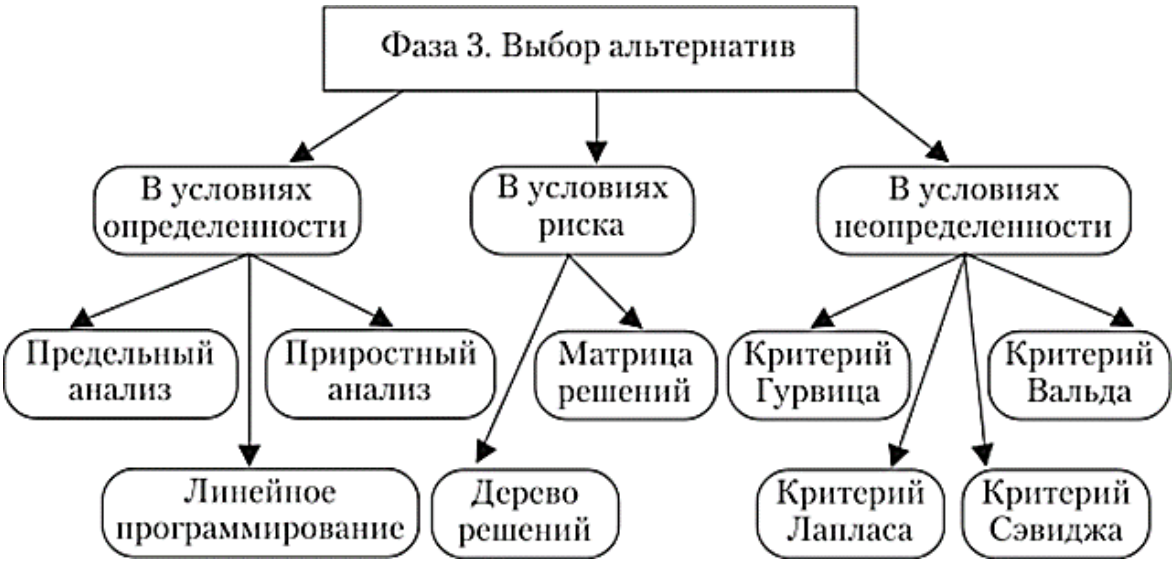

Рисунок 2.4 – Структурная схема фазы выбора альтернатив

Условия определенности представляют собой такие условия принятия решений (состояние знаний о сущности явлений), когда ЛПР заранее может определить результат (исход) каждой альтернативы, предлагаемой для выбора. Такая ситуация характерна для тактических краткосрочных решений. В этом случае ЛПР располагает подробной информацией, т.е. исчерпывающими знаниями о ситуации для принятия решения.

Условия риск характеризуются таким состоянием знания о сущности явления, когда ЛПР известны вероятности возможных последствий реализации каждой альтернативы. Условия риска и неопределенности характеризуются так называемыми условиями многозначных ожиданий будущей ситуации во

внешней среде. В этом случае ЛПР должен сделать выбор альтернативы, не имея точного представления о факторах внешней среды и их влияния на результат. В этих условиях исход, результат каждой альтернативы представляет собой функцию условий — факторов внешней среды (функцию полезности), который не всегда способен предвидеть ЛПР. Для предоставления и анализа результатов выбранных альтернативных стратегий используют матрицу решений, называемую также платежной матрицей.

Условия неопределенности представляют собой такое состояние окружающей среды (знания о сущности явлений), когда каждая альтернатива может иметь несколько результатов, а вероятность возникновения этих исходов неизвестна. Неопределенность среды принятия решения зависит от соотношения между количеством информации и ее достоверностью. Чем неопределеннее внешнее окружение, тем труднее принимать эффективные решения. Среда принятия решения зависит также от степени динамики, подвижности среды, т.е. скорости происходящих изменений условий принятия решения. Изменение условий может происходить как вследствие развития организации, т.е. приобретения ею возможности решать новые проблемы, способности к обновлению, так и под влиянием внешних по отношению к организации факторов, которые не могут регулироваться организацией [8].

Следующая фаза, реализация решений. При реализации решений применяют методы планирования, организации и контроля выполнения решений (рисунок 2.5).

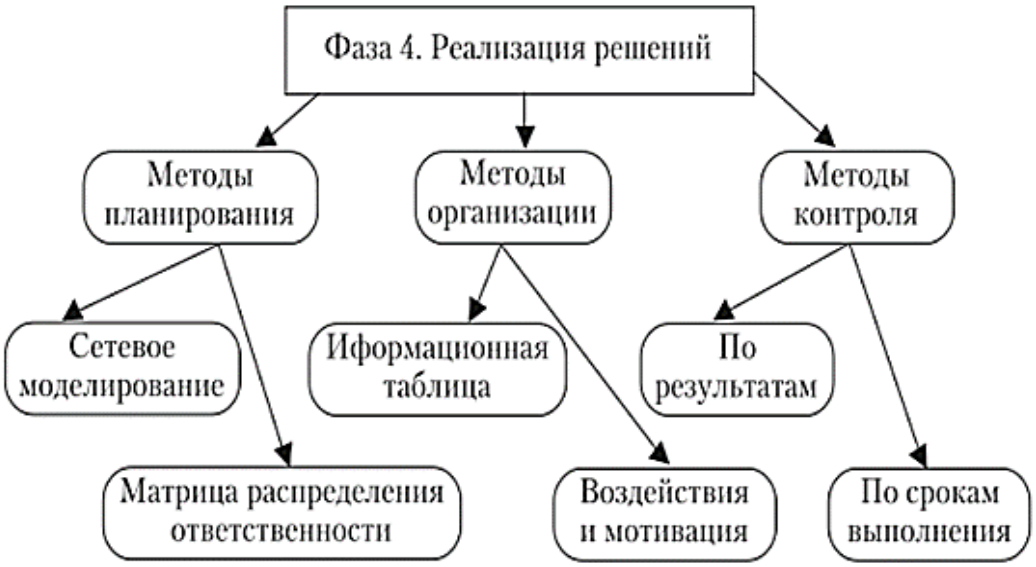

Рисунок 2.5 – Структурная схема фазы реализации решений

2.4 Описание используемого метода

Исходя из вышеописанных классификаций методов, одним из вариантов ситуации принятия решения является так называемая критериальная постановка. В этом случае лицо, принимающее решение выбирает лучшие из альтернатив для достижения определенной цели. Но соответствие цели оценивается не

непосредственно, а путем удовлетворения набору критериев, обладанием рядом свойств.

Данный метод применяется для многих задач:

- сравнительный анализ объектов (многокритериальное ранжирование);
- многокритериальный выбор лучшего объекта (лучшей альтернативы);
- распределение ресурсов между проектами;
- проектирование систем по количественным и качественным характеристикам [9].

Шаги метода анализа иерархии имеют следующий вид:

1 Происходит сравнение альтернативы между собой не с точки зрения достижения цели, а с точки зрения удовлетворения конкретным критериям. Кроме того, необходимо сравнить между собой значимость критериев для конкретной цели. (рисунок 2.6)

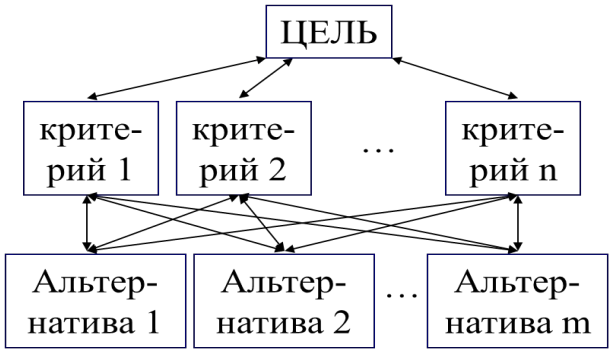

Рисунок 2.6 – Иерархическая структура ситуации принятия решения

Цель составляет высший уровень иерархии (уровень 1). На этом уровне может находиться лишь один объект. На следующих вниз уровнях находятся критерии. По системе этих критериев оцениваются сравниваемые объекты (называемые «альтернативами»). Альтернативы располагаются на самом нижнем уровне.

2 Вынесение экспертных суждений на каждом уровне иерархии по парным сравнениям: критерии сравниваются попарно по отношению к цели, альтернативы — попарно по отношению к каждому из критериев. Соответственно заполняются матрицы парных сравнений: одна — для критериев, n матриц — для альтернатив; здесь n — количество критериев. (рисунок 2.7)

	K1	K2	…	Kn
A1				
A2				
…				
An				

Рисунок 2.7 – Матрица попарных сравнений

Операция парного сравнения: два объекта, находящихся на одном уровне сравниваются по своей относительной значимости для одного объекта высшего уровня. Если критерий имеет определенную числовую меру, например, масса, производительность, цена, то в качестве результата оценки удобно взять отношения соответствующих характеристик в некоторой шкале отношений. Если критерий не имеет принятой меры, то сравнение в МАИ проводится с использованием специальной «шкалы относительной важности», приведенной в таблице 2.1.

Таблица 1.1 – Шкала относительной важности

Степень предпочтения	Определение	Критерий
1	Равная предпочтительность	Две альтернативы одинаково предпочтительны с точки зрения цели
2	Слабая степень предпочтения	Опыт эксперта позволяет считать одну из альтернатив немного предпочтительнее другой
3	Степень наибольшего предпочтения	Опыт эксперта позволяет считать одну из альтернатив явно предпочтительнее другой
4	Сильное предпочтение	Опыт эксперта позволяет считать одну из альтернатив гораздо предпочтительнее другой
5	Абсолютное предпочтение	Очевидность подавляющей предпочтительности одной альтернативы над другой имеет неоспоримое подтверждение

Данная шкала является упрощением шкалы Саати для предоставления большего удобства выражения предпочтения критериев для пользователя во время работы в приложении, чтобы можно было проще присвоить важность критерия и не вызвать каких-либо спутанных суждений. Числа из этой шкалы используются, чтобы показать, во сколько раз элемент с большей оценкой

предпочтительности доминирует элемент с меньшей оценкой относительно общего для них критерия или свойства [7].

2.5 Анализ информационных потоков разрабатываемой информационной системы

Залогом успешной реализации проекта по разработке программного обеспечения (ПО) является адекватным образом построенная модель разрабатываемой системы. Построение модели представляет собой достаточно трудоемкую и длительную по времени исполнения задачу, требующую привлечения в проект высококвалифицированных специалистов. Периодическое внесение изменений в структуру проекта дополнительно усложняет задачу и приводит к удорожанию проекта в целом. Использование CASE-средств (Computer Aided Software Engineering) позволяет автоматизировать процессы различных стадий жизненного цикла ПО (ЖЦ ПО). Именно благодаря CASE-средствам разработчики информационных систем (ИС) получили возможность представлять создаваемую систему в наглядной – графической – форме, структурировать процесс разработки модулей системы, облегчить подготовку проектной документации. В настоящее время существует множество различных CASE-средств, каждое из которых способно решать определенный круг задач в области проектирования ИС. Отдельные CASE-средства ориентированы на решение отдельных задач, например, графические инструменты описания моделей ИС в виде совокупности иерархически взаимосвязанных диаграмм, а полностью интегрированные средства представляют собой комплекс средств, поддерживающих полный ЖЦ ПО. Выбор CASE-средства зависит от множества факторов, формирующих задание на разработку конкретной ИС, и может быть осуществлен на основе сведений о характеристиках уже существующих CASE-средств [26].

Остановимся на нескольких вариантах достаточно развитых CASE-средств. Таковыми являются такие CASE-средства, как MS Visio, BPWIN и Ramus. Проведем небольшой сравнительный анализ данных CASE-средств.

MS Visio является самым простым и доступным средством моделирования процессов в использовании. Данная программа относится к семейству программ MS Office, благодаря чему имеет всем знакомый пользовательский интерфейс, а также очень легко интегрируется с любыми приложениями MS Office. MS Visio поддерживается такие форматы для описания бизнес-процессов, как IDEF и UML.

BPWIN отличается большими возможностями анализа и достаточной простотой использования. Функциональность BPWIN позволяет следить пользователю за согласованностью модели, а также обеспечивает проверку ее целостности.

Данное CASE-средство позволяет производить коррекцию ошибок при моделировании процессов, кроме того, BPWIN имеет возможность поддерживания

пользовательских свойств, которые применяются к элементам диаграммы. Главным минусом данной системы является лежащий в ее основе стандарт IDEF, который вводит жесткие ограничение при построении моделей.

Ramus – средство, предназначенное для построения или реорганизации систем управления предприятием, например, проекты внедрения процессного управления или проекты по реинжинирингу различных бизнес-процессов и тому подобное. Для полной реализации своих преимуществ данное CASE-средство необходимо использовать в достаточно больших организациях.

Главными возможностями Ramus являются:

− создание моделей процессов согласно методологиям IDEF0, IDEF3, DFD;

− разработка систем кодирования предприятия с использованием внутренних перекрестных связей;

− формирование отчетности по созданным моделям;

− генерация сайта, в котором предоставляется доступ к данным моделей процессов через web интерфейс [21].

В таблице 2.2 представлен сравнительные анализ рассматриваемых CASE-средств.

Таблица 2.2 – Сравнительный анализ рассматриваемых CASE-средств

Возможности	MS Visio	BPWIN	Ramus
Поддерживаемые стандарты	IDEF0, UML	IDEF0, IDEF3, DFD	IDEF0, IDEF3, DFD
Формат представления моделей	Не регламентируется	Стандартный бланк IDEF с возможностью его отключения	Стандартный бланк IDEF с возможностью его отключения
Генерация отчетов	Создание отчетов по UDP с использованием Visio Report	Возможность визуальной настройки отчетов при помощи RPT Win	Встроенный генератор отчетов, с гибкой структурой
Сложность разработки нестандартных отчетов	Сложно	Просто	Просто
Ограничение на количество объектов на диаграмме	В зависимости от используемого стандарта	до 8 объектов	В зависимости от используемого стандарта
Удобство работы по созданию моделей	Простая панель управления, есть выравнивание объектов, есть undo	Простая панель управления, есть выравнивание объектов, нет undo	Простая панель управления, есть выравнивание объектов, есть undo
Кроссплатформен	Присутствует	Отсутствует	Присутствует

ность			
Расчет стоимостных затрат	Присутствует	Отсутствует	Присутствует

Таким образом, используя метод экспертных оценок, можно сделать вывод, что лучшим CASE-средством является Ramus.

Далее перейдем к созданию диаграммы IDEF0 в выбранном CASE-средстве. Методология IDEF0 представляет собой совокупность методов, правил и процедур, предназначенных для построения функциональной модели объекта какой-либо предметной области. Функциональная модель IDEF0 отображает функциональную структуру объекта, т.е. производимые им действия и связи между этими действиями.

IDEF0 может быть использована для моделирования широкого класса систем. Для новых систем применение IDEF0 имеет своей целью определение требований и указание функций для последующей разработки системы, отвечающей поставленным требованиям и реализующей выделенные функции. Применительно к уже существующим системам IDEF0 может быть использована для анализа функций, выполняемых системой, и отображения механизмов, посредством которых эти функции выполняются.

Модель в IDEF0 представлена совокупностью иерархически упорядоченных и логически связанных диаграмм, а также текста документации и словарей, связанных друг с другом с помощью перекрестных ссылок [11].

Основу методологии IDEF0 составляет графический язык описания бизнес-процессов. Модель в нотации IDEF0 представляет собой совокупность иерархически упорядоченных и взаимосвязанных диаграмм. Каждая диаграмма является единицей описания системы и располагается на отдельном листе.

Можно выделить четыре типа диаграмм:

1 контекстную диаграмму Аз0 (в каждой модели может быть только одна контекстная диаграмма);

2 диаграммы декомпозиции (в том числе диаграмма первого уровня декомпозиции А0, раскрывающая контекстную);

3 диаграммы дерева узлов;

4 диаграммы только для экспозиции (FEO) [12].

Контекстная диаграмма представляет собой самое общее описание системы и ее взаимодействия с внешней средой. После описания системы в целом проводится разбиение ее на подсистемы. Этот процесс называется функциональной декомпозицией, а диаграммы, которые описывают каждый фрагмент, называются диаграммами декомпозиции. Каждый компонент модели может быть декомпозирован на другой диаграмме. Каждая диаграмма иллюстрирует "внутреннее строение" блока на родительской диаграмме. После декомпозиции контекстной диаграммы (т.е. получения диаграммы А0) проводится декомпозиция каждого блока диаграммы А0 на более мелкие фрагменты и так далее, до достижения нужного уровня подробности описания. После каждого сеанса декомпозиции проводятся сеансы экспертизы: эксперты

предметной области (обычно это интервьюируемые аналитиками сотрудники предприятий) указывают на соответствие реальных бизнес-процессов созданным диаграммам. Найденные несоответствия исправляются, и только после прохождения экспертизы без замечаний можно приступать к следующему сеансу декомпозиции. Так достигается соответствие модели реальным бизнес-процессам на любом и каждом уровне модели. Синтаксис описания системы в целом и каждого ее фрагмента одинаков во всей модели.

Диаграмма дерева узлов показывает иерархическую зависимость работ, но не взаимосвязи между работами. Диаграмм деревьев узлов может быть в модели сколько угодно, поскольку дерево может быть построено на произвольную глубину и не обязательно с корня.

Диаграммы для экспозиции (FEO) строятся для иллюстрации отдельных фрагментов модели, для иллюстрации альтернативной точки зрения, либо для специальных целей.

Правила IDEF0 включают:

1 ограничение количества блоков на каждом уровне декомпозиции (правило 3-6 блоков);

2 связность диаграмм (номера блоков);

3 уникальность меток и наименований (отсутствие повторяющихся имен);

4 синтаксические правила для графики (блоков и дуг);

5 разделение входов и управлений (правило определения роли данных);

6 отделение организации от функции, т.е. исключение влияния организационной структуры на функциональную модель [13].

Контекстный уровень диаграммы информационной системы замены поддержки принятия решений по замене сетевого оборудования организации представлена на рисунке 2.8.

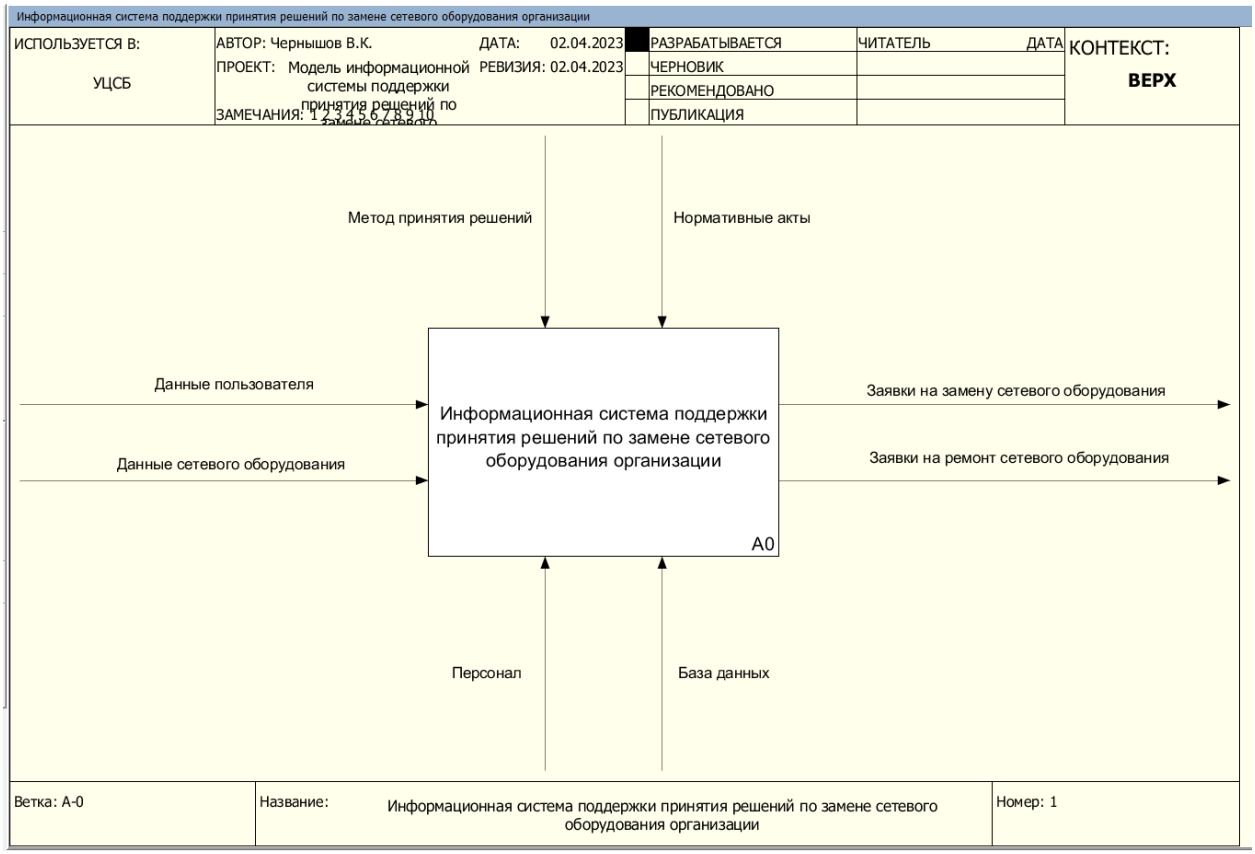

Рисунок 2.8 – Контекстная диаграмма

Далее этот блок детализируется на другой диаграмме с помощью нескольких блоков, соединенных интерфейсными дугами. Декомпозиция - разделение моделируемой функции на функции компоненты. Диаграмма декомпозиции представлена на рисунке 2.9.

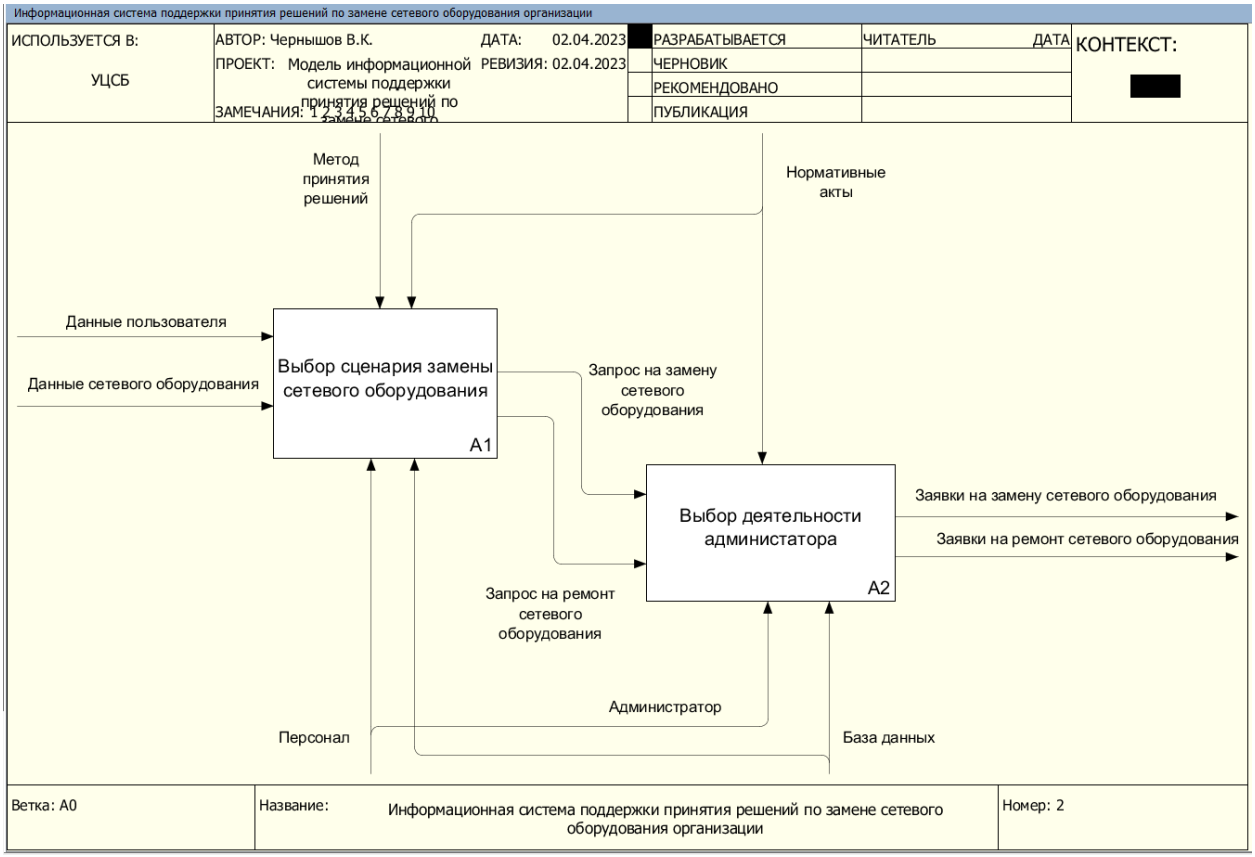

Рисунок 2.9 – Диаграмма декомпозиции A0

Т.к. основная деятельность — это определение решения по замене сетевого оборудования, то именно это и покажем в качестве диаграммы декомпозиции второго уровня рисунок 2.10.

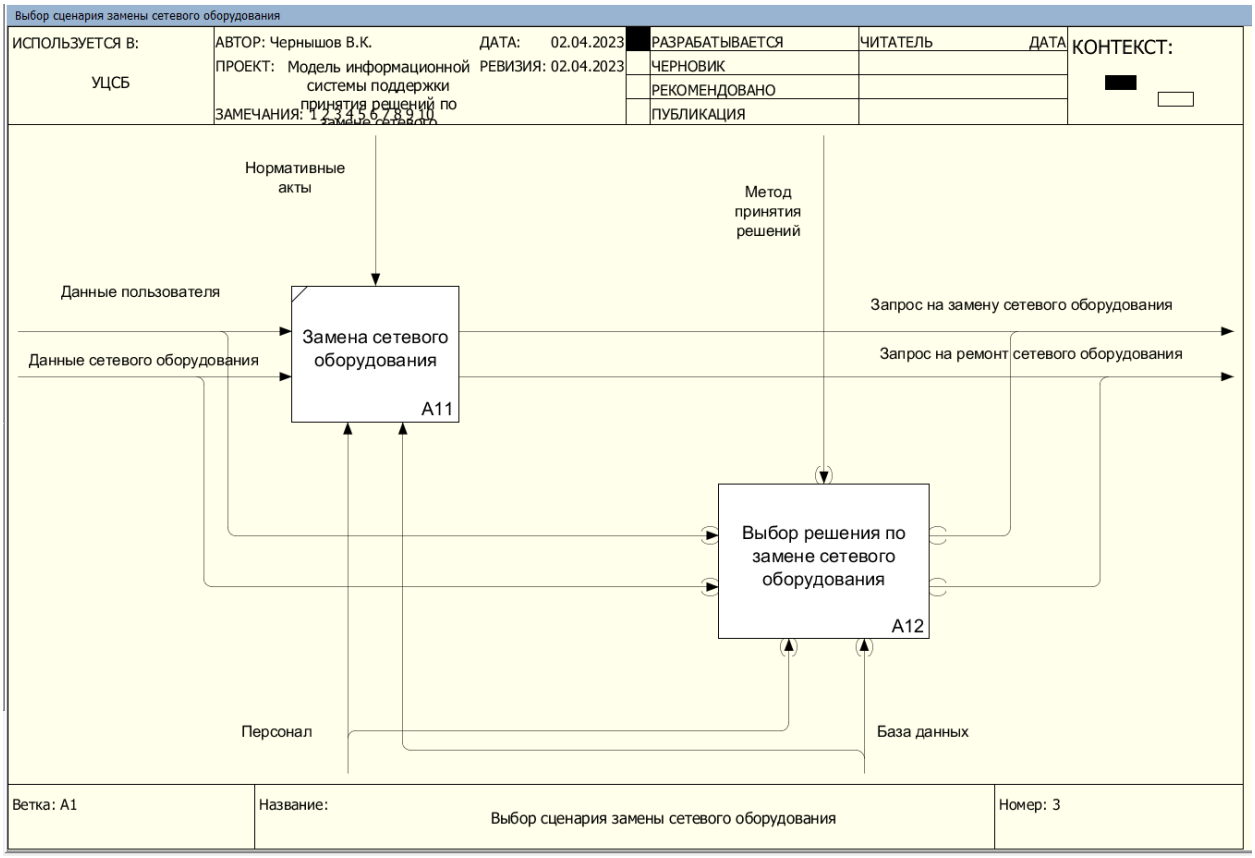

Рисунок 2.10 – Диаграмма декомпозиции функционального блока A1

На рисунке 2.11 отображена деятельность администратора системы.

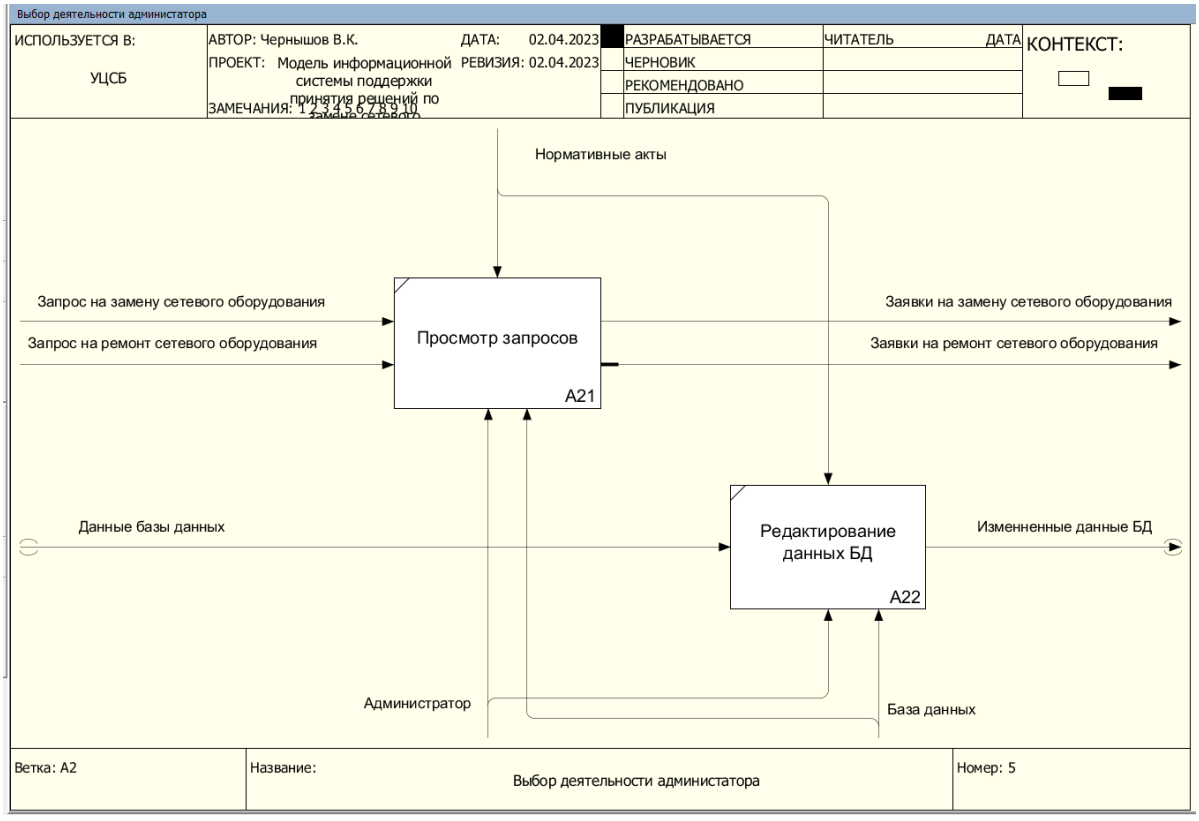

Рисунок 2.11 – Диаграмма декомпозиции функционального блока A2

Диаграммы потоков данных (DFD - Data Flow Diagram) - основные средства моделирования функциональных требований проектируемой системы. С их помощью эти требования разбиваются на функциональные компоненты (процессы) и представляются в виде сети, связанной потоками данных. Главная цель таких средств - продемонстрировать, как каждый процесс преобразует свои входные данные в выходные, а также выявить отношения между этими процессами [14].

DFD – это нотация, предназначенная для моделирования информационный систем с точки зрения хранения, обработки и передачи данных [15].

Пример схемы информационных потоков в виде диаграммы потоков данных (DFD) представлен на рисунке 2.12.

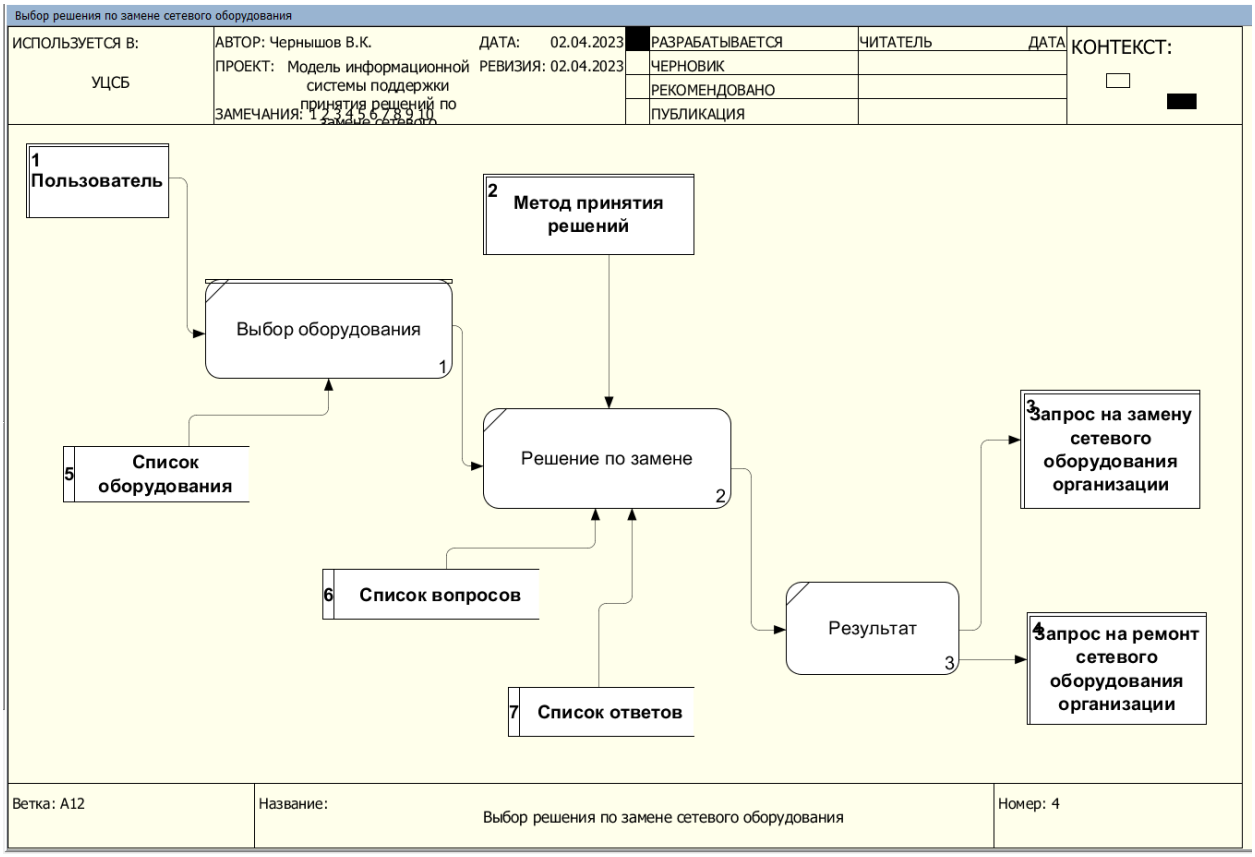

Рисунок 2.12 – Диаграмма потока данных

2.6 Объектно-ориентированное моделирование информационной системы

Проектирование каких-либо систем уделяет основное внимание правильному и эффективному структурированию сложных систем. Объектно-ориентированное проектирование (далее – ООП) можно определить следующим образом. ООП – это методология проектирования, соединяющая в себе процесс объектной декомпозиции и приемы представления логической и физической, а также статической и динамической моделей проектируемой системы [16].

Из данного определения следует, что ООП основывается на объектной декомпозиции и использует многообразие приемов представления моделей, отражающих логическую (классы и объекты) и физическую (модули и процессы) структуры системы, а также статические и динамические аспекты.

Именно объектно-ориентированная декомпозиция является основополагающим принципом ООП. При этом требования к проектируемой системе представляются с точки зрения классов и объектов, выявленных в предметной области. Логическая структура системы также отображается абстракциями в виде классов и объектов [17].

Объектно-ориентированная разработка программного обеспечения связана с применением объектно-ориентированных технологий. Обычно эти объектно-ориентированные методологии поддерживаются инструментальными программными средствами, но и без таких средств они полезны, так как позволяют хорошо понять различные аспекты и свойства разрабатываемой программной системы, что в последующем существенно облегчает ее реализацию, тестирование, сопровождение, разработку новых версий и более существенную модификацию [18].

При разработке объектно-ориентированных моделей информационной системы важно правильно выбрать инструментальные средства.

Инструментальные средства разработки информационной системы – это набор аппаратно-программных средств, которые обеспечивают полный цикл проектирования программного обеспечения [22].

На данный момент существует большое количество CASE-средств, которые используют при моделировании нотацию UML. Например: Rational Rose, Umbrello, Edraw Max, UML Designer, Altova UModel, Microsoft Visio, Umple, Visual Paradigm, StarUML, WhitestarUML, GenMyModel, Diagramo, Astah и др.

Чтобы выбрать наиболее подходящее CASE-средство, необходимо провести сравнительный анализ именно тех программных продуктов, с которыми имелся опыт работы. Следовательно, сравниваться будут StarUML, Rational Rose и Microsoft Visio.

Проведем сравнительный анализ функциональных возможностей указанных выше CASE-средств и сведем результаты сравнения в таблицу 2.3.

Таблица 2.3 – Сравнительный анализ функциональных возможностей средств ООП

№	Инструментальная среда Возможности	StarUML	Rational Rose	Microsoft Visio
1	Дистрибуция в свободном доступе	Да	Да	Нет. Необходима подписка Office 365.
2	Поддержка стандарта	Да	Нет	Да

	UML 1.4, нотации UML 2.0 на основе метамодели			
3	Поддержка MDA	Да	Нет	Нет
4	Возможность совместной работы	Нет	Да. При помощи IBM Rational Suite и Visual SourseSafe.	Нет
5	Расширяемость	Да. Поддержка Microsoft COM.	Нет	Да. Поддержка Microsoft COM.
6	Программная функция проверки модели	Да	Да	Нет
7	Интерфейс пользователя	Простой интерфейс, есть выравнивание объектов, функция отмены предыдущих действий.	Сложный интерфейс, отсутствует выравнивание, однократная отмена предыдущих действий.	Простой интерфейс, есть выравнивание объектов, есть многократная отмена предыдущих действий.
8	Полезные дополнения	Да. Импорт файлов в Rational Rose, конвертация исходных текстов в модели, генерация исходных текстов на языках программирования.	Да. Импорт модели базы данных в SQL-запрос.	Нет
9	Возможность проведения декомпозиции	Да. Имеется неограниченное количество декомпозиций. Переход от одной нотации к другой.	Да. Имеется неограниченное количество декомпозиций.	Да. Имеется неограниченное количество декомпозиций.
10	Открытый формат программной модели	Да	Нет	Да

Таким образом, проведя сравнительный анализ CASE-средств для объектно-ориентированного моделирования, было выявлено, что StarUML является наилучшей из рассматриваемых.

Далее необходимо выполнить построение UML-диаграмм.

Диаграммы прецедентов составляют модель прецедентов (вариантов использования, use-cases). Прецедент — это функциональность системы, позволяющая пользователю получить некий значимый для него, ощутимый и измеримый результат. Каждый прецедент соответствует отдельному сервису, предоставляемому моделируемой системой в ответ на запрос пользователя, т. е. определяет способ использования этой системы. Именно по этой причине use-cases, или прецеденты, часто в русской терминологии фигурируют как варианты использования. Варианты использования чаще всего применяются для спецификации внешних требований к проектируемой системе или для спецификации функционального поведения уже существующей системы. Кроме этого, варианты использования неявно описывают типичные способы взаимодействия пользователя с системой, позволяющие корректно работать с предоставляемыми системой сервисами [19].

Диаграмма прецедентов для выбранной нами информационной системы поддержки принятия решений по замене сетевого оборудования организации, выполнена в программе StarUML и представлена на рисунке 2.13.

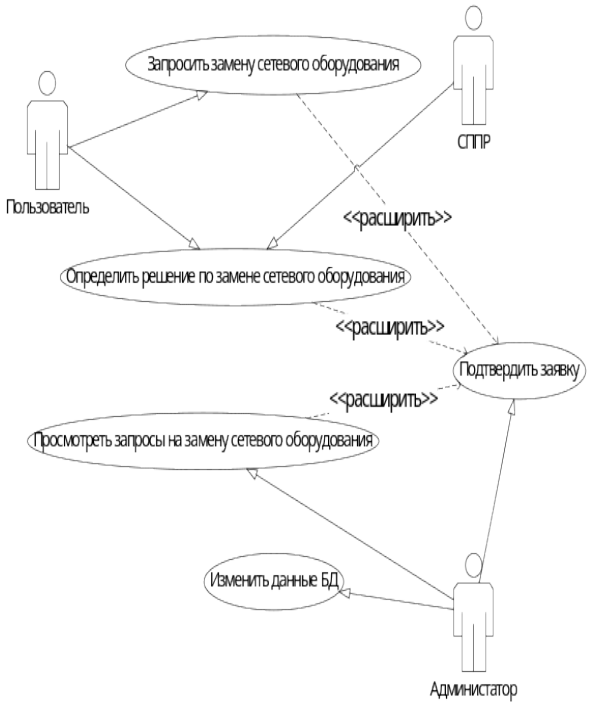

Рисунок 2.13 – Диаграмма прецедентов

Целью диаграмм взаимодействия является визуализация интерактивного поведения системы. Визуализация взаимодействия — сложная задача. Следовательно, решение состоит в том, чтобы использовать различные типы моделей, чтобы охватить различные аспекты взаимодействия.

Диаграмма взаимодействия наглядно отображает временной аспект взаимодействия. Она имеет два измерения. Одно измерение (слева-направо) указывает на порядок вовлечения экземпляров сущностей во взаимодействие. Крайним слева на диаграмме отображается экземпляр актера или объект, который является инициатором взаимодействия. Правее отображается другой экземпляр сущности, который непосредственно взаимодействует с первым и т.д. Второе измерение (сверху-вниз) указывает на порядок обмена сообщениями. Начальному моменту времени соответствует самая верхняя часть диаграммы.

Так же диаграмму взаимодействий можно назвать UML диаграммой в нотации, которой взаимодействие элементов рассматривается в информационном аспекте их коммуникации. Другими словами, взаимодействующие объекты обмениваются между собой некоторой информацией. При этом информация представляет собой законченные сообщения.

На рисунке 2.14 изображена диаграмма взаимодействия «Запрос на замену сетевого оборудования»

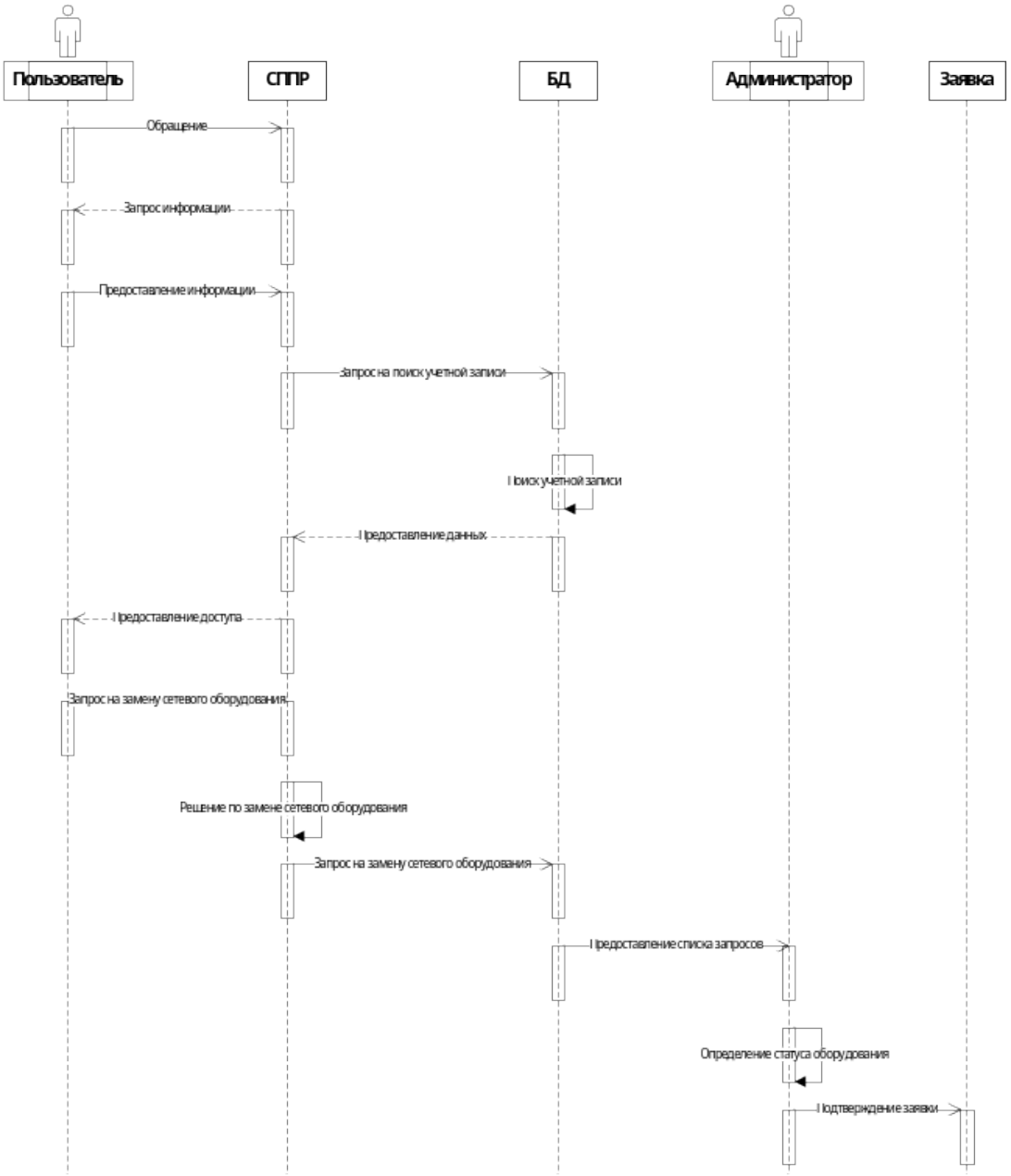

Рисунок 2.14 – Диаграмма взаимодействия

Диаграмма активностей – отражает динамические аспекты поведения системы, то есть отображает потоки работ во взаимосвязанных вариантах использования. По существу, эта диаграмма представляет собой блок – схему, которая наглядно показывает, как поток управления переходит от одной деятельности к другой. Диаграммы активностей позволяют реализовать в языке UML особенности процедурного и синхронного управления, обусловленного завершением внутренних действий и деятельности. Данная диаграмма была выполнена в программе StarUML на рисунке 2.15.

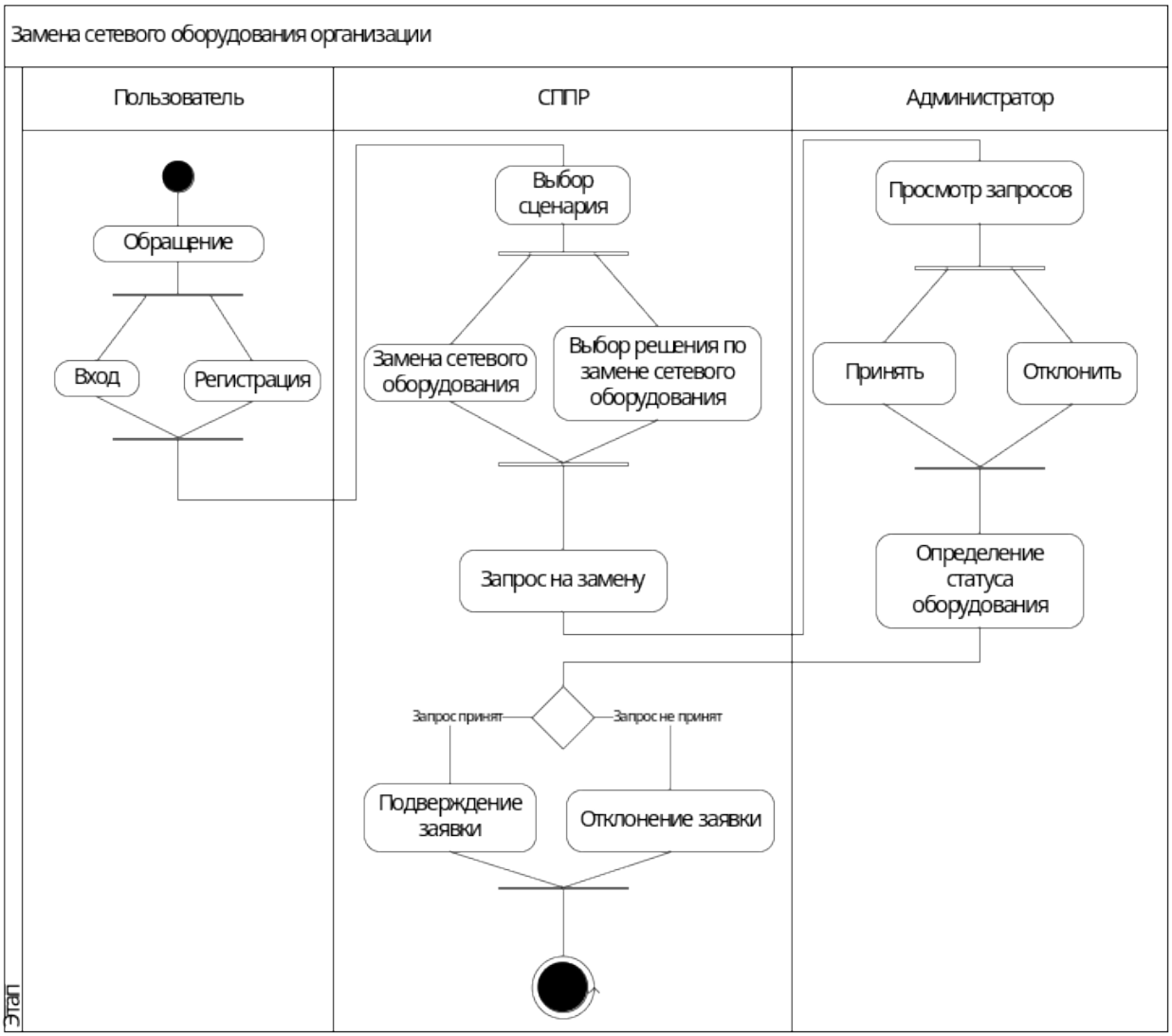

Рисунок 2.15 – Диаграмма активностей

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.

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 должны относиться к реальной разработке и составлению диаграммы (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:

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

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

3) 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 нем те общие свойства, которые не применяются на выбранном языке программирования [20].

При создании диаграммы классов важным моментом является реализация правильных связей между компонентами диаграммы. В каждом классе должен иметься первичный и вторичный ключ и стрелкой показана связь между классами. Связи могу быть следующих видов ассоциация, агрегация, наследование. Пример реализации диаграммы классов представлен на рисунке 2.16.

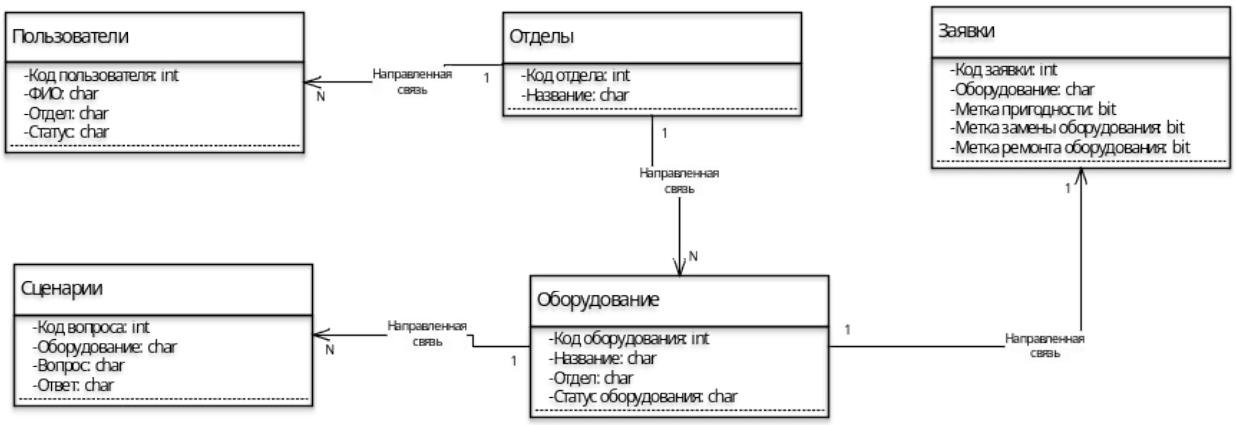

Рисунок 2.16 – Диаграмма классов

Диаграммы компонентов используются для визуализации организации компонентов системы и зависимостей между ними. Они позволяют получить высокоуровневое представление о компонентах системы.

Компонентами могут быть программные компоненты, такие как база данных или пользовательский интерфейс; или аппаратные компоненты, такие как схема, микросхема или устройство; или бизнес-подразделение, такое как поставщик, платежная ведомость или доставка.

Компонентные диаграммы:

1) Используются в компонентно-ориентированных разработках для описания систем с сервис-ориентированной архитектурой;

2) Показать структуру самого кода;

3) Может использоваться для фокусировки на отношениях между компонентами, скрывая при этом детализацию спецификации;

4) Помощь в информировании и разъяснении функций создаваемой системы заинтересованным сторонам.

На рисунке 2.17 изображена диаграмма компонентов.

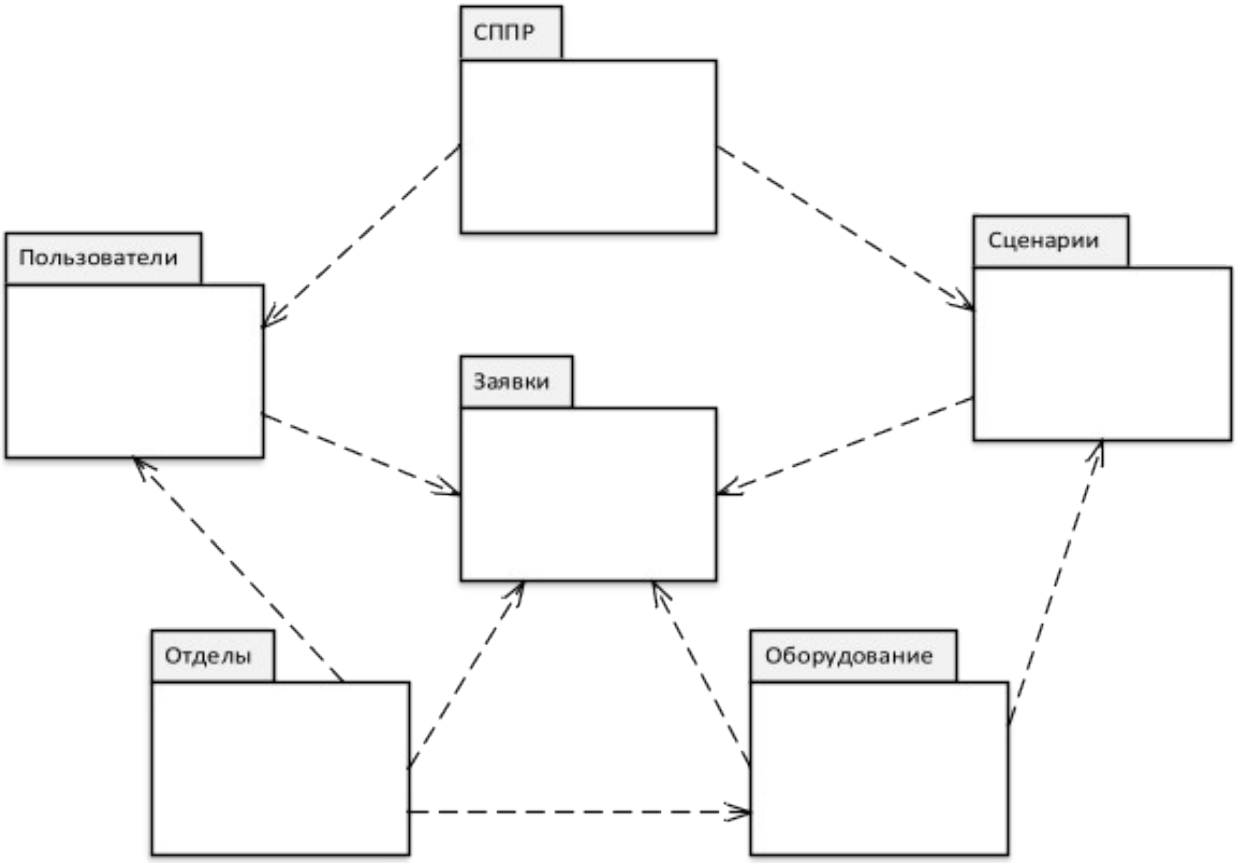

Рисунок 2.17 – Диаграмма компонентов

Диаграммы размещения используются для визуализации аппаратных процессоров/узлов/устройств системы, каналов связи между ними и размещения программных файлов на этом аппаратном обеспечении.

Диаграммы размещения обычно включают в себя узлы, а также связи зависимости и ассоциации; подобно всем прочим диаграммам, могут содержать примечания и ограничения. На диаграммах размещения бывают представлены артефакты, каждый из которых должен располагаться на каком-нибудь узле, а кроме того, пакеты или подсистемы, – и те, и другие используются для группирования элементов модели в крупные блоки. Иногда бывает полезно поместить в диаграмму объектов еще и экземпляры, особенно если вы хотите визуализировать один экземпляр из семейства топологии аппаратных средств.

Во многих отношениях диаграмма размещения является разновидностью диаграмм классов, фокусирующей внимание прежде всего на системных узлах.

Диаграммы размещения используются для моделирования статического вида системы с точки зрения размещения. Это представление в первую очередь обращено на распределение, поставку и установку частей, из которых состоит физическая система.

При моделировании статического представления системы с точки зрения размещения диаграммы размещения используются, как правило, в трех случаях:

1) Для моделирования встроенных систем. Встроенной (embedded) системой называется аппаратный комплекс, взаимодействующий с физическим миром, в котором велика роль программного обеспечения. Встроенные системы управляют двигателями, приводами и дисплеями, а сами управляются внешними воздействиями, например, датчиками температуры и перемещения. Диаграмму размещения можно использовать для моделирования устройств и процессоров, из которых состоит встроенная система.

2) Моделирование клиент-серверных систем. Клиент-серверная (client/server) система – это типичный пример архитектуры, где основное внимание уделяется четкому распределению обязанностей между интерфейсом пользователя, существующим на клиенте, и хранимыми данными системы, существующими на сервере. Клиент-серверные системы находятся на одном конце спектра распределенных систем и требуют от вас принятия решений о том, как связать клиенты и серверы сетью, а также о том, как физически распределены программные артефакты между узлами. Диаграммы размещения позволяют моделировать топологию такой системы.

3) Моделирование полностью распределенных систем. На другом конце спектра распределенных систем находятся те из них, которые распределены широко или даже глобально (fully distributed) и охватывают серверы различных уровней. Часто на таких системах устанавливаются разные версии программных компонентов, часть которых даже мигрирует с одного узла на другой. Проектирование подобной системы требует решений, которые допускают непрерывное изменение системной топологии. Диаграммы размещения можно использовать для визуализации текущей топологии и распределения артефактов системы, чтобы можно было осмысленно говорить о влиянии на нее различных изменений.

Диаграмма размещения для информационной системы библиотеки вуза представлена на рисунке 2.18.

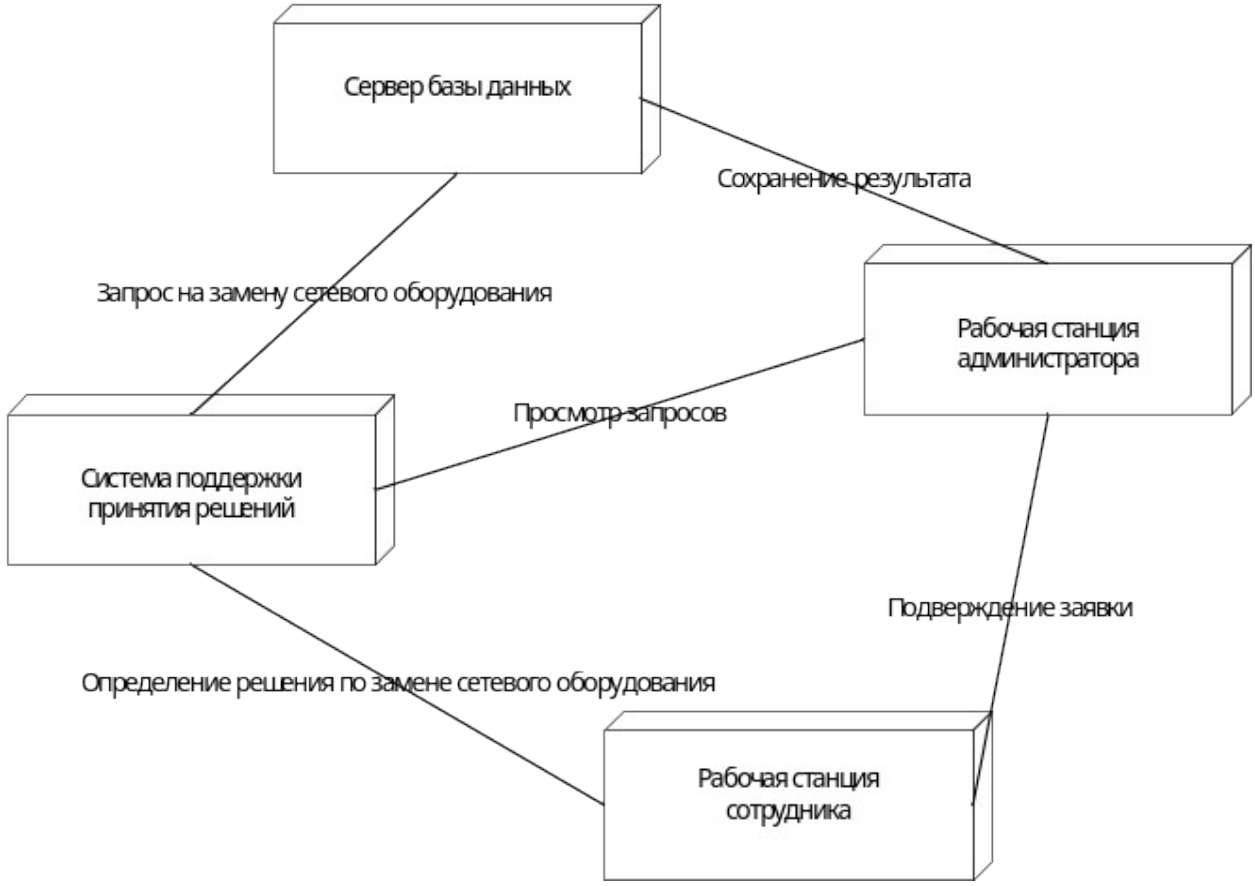

Рисунок 2.18 – Диаграмма размещения

3 Разработка информационной системы

3.1 Выбор СУБД

База данных – это упорядоченный набор данных, который предусматривает все стандартные правила хранения и обработки информации, а также организован по установленным требованиям [23].

Для взаимодействия и управления базой данных используются СУБД (системы управления базами данных).

СУБД – это программное обеспечение, с помощью которого пользователи могут определять, создавать и поддерживать базу данных, а также осуществлять к ней контролируемый доступ [24].

В настоящее время значительное количество современных СУБД базируются на реляционной модели данных. Реляционная модель данных – это совокупность данных, которые представлены в виде связанных между собой таблиц. Для получения информации из таблиц базы данных используется декларативный язык программирования (запросов) SQL. К основным функциям СУБД можно отнести:

– обеспечение быстрого доступа к данным;
– управление транзакциями;
– поддержка целостности данных;
– гарантирование безопасности данных;
– реализация языка запросов к данным;
– ведение журнала изменений, произошедших с базой данных [25].

Как и с CASE-средствами, для выбора наиболее подходящей СУБД необходимо провести сравнительный анализ функциональных возможностей, который приведен в таблице 3.1.

Таблица 3.1 – Сравнительный анализ функциональных возможностей

СУБД	Тип	Разработчик	Операционная система	Лицензия	Исходный код	Поддержка
Oracle DB	Мультимодельная	Oracle Corporation	Linux, MS Windows, Oracle Solaris, IBM AIX	Коммерческая	Закрытый	Платная
MySQL	Реляционная	Oracle Corporation	Linux, MS Windows, Oracle Solaris, Free	GNU GPL и коммерческая	Открытый	Платная

			BSD, macOS			
SQL Server	Реляционная	Microsoft	MS Windows, Linux	Коммерческая	Закрытый	Бесплатная

На основе сравнительного анализа можно сделать вывод, что СУБД MS SQL Server является наиболее подходящей для работы с разрабатываемой информационной системой.

3.2 Проектирование базы данных

Проектирование базы данных СППР по замене сетевого оборудования организации. В результате проведенного анализа предметной области легко перечислить все основные сущности этой БД.

В реляционную модель проектированной БД будут входить следующие таблицы (сущности): «Тестовые анкеты», «Вопросы», «Опросы», «Результаты опросов», «Пользователи» (Таблицы 3.2-3.6).

Таблица 3.2 – Список атрибутов таблицы «Тестовые анкеты»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	idTestWithQuestion	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому туру. Это целое число. Т.е. для идентификации каждого тура будет применяться определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
ВК (внешний ключ)	idQuestion	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это

		атрибут таблицы «Вопросы». С помощью внешнего ключа будет определено описание вопросов для анкеты.
ВК (внешний ключ)	idTest	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы «Опросы». С помощью внешнего ключа будет определено описание опроса.

Таблица 3.3 – Список атрибутов таблицы «Вопросы»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	idQuestion	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому туру. Это целое число. Т.е. для идентификации каждого тура будет применяться определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Type	
	Question	
	Answer	

Таблица 3.4 – Список атрибутов таблицы «Опросы»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	idTest	Ключевое поле,

		предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому туру. Это целое число. Т.е. для идентификации каждого тура будет применяться определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
ВК (внешний ключ)	idUser	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы «Пользователи». С помощью внешнего ключа будет определено описание пользователя.
	About	

Таблица 3.5 – Список атрибутов таблицы «Результаты опросов»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	idReultTest	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому туру. Это целое число. Т.е. для идентификации каждого тура будет применяться

		определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
ВК (внешний ключ)	idUser	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы «Пользователи». С помощью внешнего ключа будет определено описание пользователя.
ВК (внешний ключ)	idTest	Внешний ключ – это атрибут отношения, который является первичным ключом другого отношения. В нашем случае это атрибут таблицы «Опросы». С помощью внешнего ключа будет определено описание опроса.
	DataTesting	
	Result	

Таблица 3.6 – Список атрибутов таблицы «Пользователи»

Ключевое поле	Название	Назначение
ПК (первичный ключ)	idUser	Ключевое поле, предназначенное для однозначной идентификации каждой записи в таблице. Представляет собой первичный ключ. Это уникальное значение, соответствующее каждому туру. Это целое число. Т.е. для идентификации каждого тура будет применяться

		определенный номер. Этот номер может быть случайным целым числом или счетчик по порядку.
	Name	
	Surname	
	Login	
	Password	
	Role	

Еще одним важнейшим этапом проектирования базы данных является даталогическое проектирование. Описание сущностей предметной области для логической модели сокращенной базы данных СППР по замене СО приведено на рисунке 3.1.

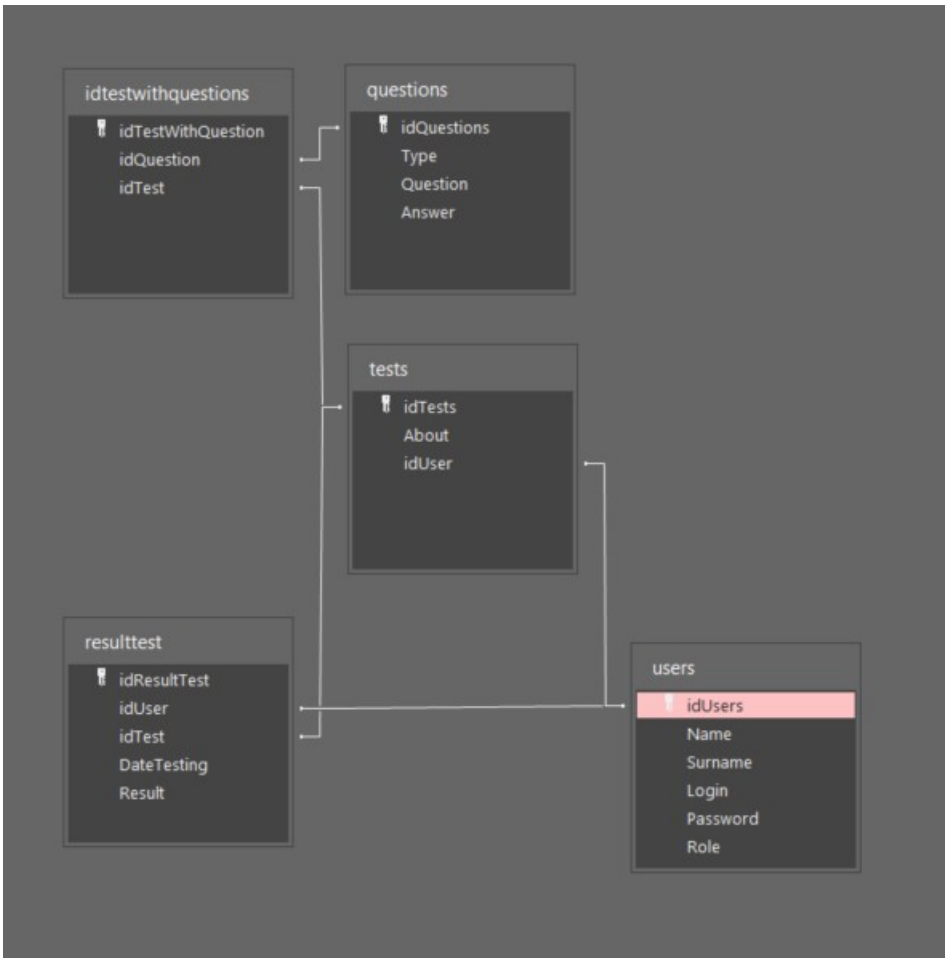

Рисунок 3.1 – Логическая модель базы данных СППР по замене СО

3.3 Выбор среды разработки

Выбор средств разработки. Для реализации поставленной задачи была выбрана среда разработки Microsoft Visual Studio 2022, а также язык программирования C#. Данный язык имеет достаточно большой набор готовых

библиотечных функций, что ускоряет написание программ, включает в себя поддержку объектно-ориентированного, структурного, а также компонентно-ориентированного программирования, а также дает возможность интерактивно конструировать вид графического интерфейса разрабатываемого приложения. Он имеет отличную поддержку компонентов, устойчив и надежен благодаря обработки исключений, сборки мусора и присутствия безопасности типов данных.

Microsoft Visual Studio 2022 – линейка продуктов, включающая интегрированную среду разработки с удобным интерфейсом. Microsoft Visual Studio является полноценным инструментальным средством, которое позволяет автоматизировать деятельность команды разработчиков.

Преимущества Visual Studio:

− среда содержит множество инструментов, которые очень хорошо работают на C#;

− Visual Studio имеет мощную поддержку навигации и рефакторинга;

− технология авто дополнения IntelliSense;

− среда имеет расширенный отладчик кода, анализ дампа памяти и статический анализ кода;

− Visual Studio дает возможность использовать все возможности платформы Microsoft, для обеспечения контроля за качеством выпускаемого продукта.

3.4 Разработка приложения

После проведения необходимого анализа, разработки СУБД и выбора среды разработки, начинается следующий шаг – разработки приложения.

Первая форма и лицо разрабатываемого ПО – аутентификация. Код приведен на рисунке 3.2.

```
Ссылок: 1
private void button2_Click(object sender, EventArgs e)
{
    if (!string.IsNullOrWhiteSpace(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox2.Text))
    {
        SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
        SqlDataAdapter adapter = new SqlDataAdapter(string.Format("select Name,SurName,Login,Role,idUsers from Users where Login = '{0}' and Password = '{1}'",textBox1.Text,textBox2.Text), connect);
        DataTable table = new DataTable();
        adapter.Fill(table);
        if(table.Rows.Count!=0)
        {
            user.ID = int.Parse(table.Rows[0]["idUsers"].ToString());
            user.Name = table.Rows[0]["Name"].ToString();
            user.SurName = table.Rows[0]["SurName"].ToString();
            user.Login = table.Rows[0]["Login"].ToString();
            user.UserRole = (Role)int.Parse(table.Rows[0]["Role"].ToString());

            Close();
        }
        else
            MessageBox.Show("Логин или пароль неверны!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    else
        MessageBox.Show("Поля логин и пароль не могут быть пустыми!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
}

Ссылок: 1
private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
{
    Registration reg = new Registration();
    reg.ShowDialog();
}
```

Рисунок 3.2 – Код авторизации

Первым делом, необходимо установить соединение с СУБД, т.к. данные всех пользователей хранятся в ней. Используя команду, изображенную на рисунке 3.3, устанавливается связь с СУБД.

```
SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
SqlDataAdapter adapter = new SqlDataAdapter(string.Format("select Name,SurName,Login,Role,
```

Рисунок 3.3 – Код подключения к СУБД

Далее происходит инициализация переменной DataTable для временного хранения вводимых данных пользователя. Так же добавляется защита от введения непустых и корректных данных, код изображен на рисунке 3.4.

```
if(table.Rows.Count!=0)
{
    user.ID = int.Parse(table.Rows[0]["idUsers"].ToString());
    user.Name = table.Rows[0]["Name"].ToString();
    user.SurName = table.Rows[0]["SurName"].ToString();
    user.Login = table.Rows[0]["Login"].ToString();
    user.UserRole = (Role)int.Parse(table.Rows[0]["Role"].ToString());

    Close();
}
```

Рисунок 3.4 – Код защиты ввода некорректных данных

Также необходимо предусмотреть отсутствие пользователя в СУБД и его добавление с помощью дополнительной формы – регистрации. Код формы регистрации изображен на рисунке 3.5.

```
Ссылок: 1
private void button2_Click(object sender, EventArgs e)
{
    SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
    SqlCommand command = new SqlCommand(string.Format("insert into users(Name,SurName,Login,Password,Role) values('{0}','{1}','{2}','{3}',1)"
    try
    {
        connect.Open();
        command.ExecuteNonQuery();
        connect.Close();
        MessageBox.Show("Регистрация прошла успешно!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch {
        MessageBox.Show("При регистрации возникла ошибка!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
    Close();
}
```

Рисунок 3.5 – Код регистрации

Для проверки удачного подключения и заполнения данных СУБД используются конструкция перехвата ошибок «try… catch».

Следующий шаг после разработки необходимых компонентов для инициализации пользователя – разработка основной формы.

Данная форма должна содержать в себе несколько функций:

- отображение данных по прохождению опроса;
- возможность перехода к опросникам;
- возможность разработки и добавления новых опросников.

Т.к. программа подразумевает собой расширенный функционал, необходимо разделить возможности пользователей по ролям:

- 0, админ;
- 1, пользователь.

В административную роль необходимо добавить следующие компоненты:

- отображение данных по прохождению опроса всех пользователей;
- возможность создавать новые вопросы и опросники.

Для этого в функцию вывода данных по опросам с СУБД добавляется условие проверяющее наличие прав администратора, код изображен на рисунке 3.6.

```
dataGridView1.DataSource = null;
SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
 if (toolStripStatusLabel1.Text == string.Format("admin admin (admin)"))
//Вывод для админа
{ SqlDataAdapter adapter = new SqlDataAdapter(string.Format("SELECT Name,About,Result,DateTesting " +
    "FROM resulttest inner join tests on resulttest.idTest=tests.idTests " +
    "inner join users on resulttest.idUser = users.idUsers"), connect);
    DataTable table = new DataTable();
    adapter.Fill(table);

    dataGridView1.DataSource = table.DefaultView;
}
```

Рисунок 3.6 – Проверка роли админа

Для пользователя в функции вывода добавляется условие «where», которое позволяет вывести результат опроса только для авторизированного в данный момент пользователя, код изображен на рисунке 3.7.

```
//Вывод для пользователя
{ SqlDataAdapter adapter = new SqlDataAdapter(string.Format("SELECT About,Result,DateTesting " +
   "FROM resulttest inner join tests on resulttest.idTest=tests.idTests inner join users on " +
   "resulttest.idUser = users.idUsers where idUsers={0}", user.ID), connect);
   DataTable table = new DataTable();
   adapter.Fill(table);

   dataGridView1.DataSource = table.DefaultView;
}
```

Рисунок 3.7 – Вывод результата опроса

Для создания и прохождения опросников вводятся 2 экземпляра:

- создатьТестToolStripMenuItem_Click;
- пройтиТестToolStripMenuItem_Click.

При создании опроса используется тип вопроса – задание с выбором ответа, код изображен на рисунке 3.8.

```
{
    //Задание с выбором ответа!
    var result = WriteDataInDB(string.Format("insert into questions(Type,Value1,Value2)" +
        " values({0},'{1}','{2}')", comboBox1.SelectedIndex, textBox4.Text, textBox3.Text));

    if (result)
        AddInListView();
}
```

Рисунок 3.8 – Код инициализации задания с выбором ответа

Для занесения опроса в СУБД используется SqlCommand, а также для проверки корректности вводимых данных используется конструкция «try… catch», код изображен на рисунке 3.8.

```
{
    SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
    SqlCommand command = new SqlCommand(commandText, connect);
    try
    {
        connect.Open();
        command.ExecuteNonQuery();
        connect.Close();
        return true;
    }
    catch {
        return false;
    }
}
```

Рисунок 3.9 – Проверка вводимых данных нового опроса в СУБД

После добавления новых опросников, формируется общий файл-отчет, код изображен на рисунке 3.10.

```
private void ReadQuestions()
{
    checkedListBox1.Items.Clear();
    list.Clear();
    SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
    SqlDataAdapter adapter = new SqlDataAdapter("select * from questions", connect);
    DataTable table = new DataTable();
    adapter.Fill(table);

    for (int i = 0; i < table.Rows.Count; i++)
    {
        list.Add(
            new Questions
            {
                CategoryObj = (CategoryQuest)int.Parse(table.Rows[i]["Type"].ToString()),
                Index = int.Parse(table.Rows[i]["idQuestions"].ToString()),
                Value1 = table.Rows[i]["Value1"].ToString(),
                Value2 = table.Rows[i]["Value1"].ToString()
            });
    }
    list.OrderBy(x => x.CategoryObj);

    foreach (var item in list)
    {
        checkedListBox1.Items.Add(string.Format("({0}) {1}",item.CategoryObj,item.Value1));
    }

}
```

Рисунок 3.10 – Код формирования нового опросника

Для считывания и итогового формирования опросника используется List<>, код изображен на рисунке 3.11.

```
{
    try
    {
        SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
        SqlCommand command = new SqlCommand(string.Format("insert into tests(About,idUser)" +
            " values('{0}',{1}); SELECT CAST(scope_identity() AS int)", textBox1.Text, user.ID), connect);
        connect.Open();
        var idTest = (int)command.ExecuteScalar();
        connect.Close();

        List<string> values = new List<string>();

        for (int i = 0; i < checkedListBox1.CheckedItems.Count; i++)
        {
            values.Add(string.Format("({0},{1})", list[checkedListBox1.CheckedIndices[i]].Index, idTest));
        }

        var result = string.Join(",", values);

        command = new SqlCommand(string.Format("insert into testwithquestions(idQuestion,idTest) values {0}", result), connect);
        connect.Open();
        command.ExecuteNonQuery();
        connect.Close();
        MessageBox.Show("Создание прошло успешно!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);
        Close();
    }
    catch
    {
        MessageBox.Show("При создании возникла ошибка!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Рисунок 3.10 – Код формирования нового опросника

Финальная форма – прохождение опроса. Для считывания вопроса с СУБД и отображения его в форме используется конструкция SqlDataAdapter и DataTable, код изображен на рисунке 3.11.

```
private void GetFullTest()
{
    string command = "SELECT * FROM testwithquestions inner join tests ON testwithquestions.idTest=tests.idTests " +
        "inner join questions on " +
        "testwithquestions.idQuestion = questions.idQuestions where tests.idTests = " + test.Index.ToString();
    SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
    SqlDataAdapter adapter = new SqlDataAdapter(string.Format("{0}",command), connect);
    DataTable table = new DataTable();
    adapter.Fill(table);

    for (int i = 0; i < table.Rows.Count; i++)
    {
        listQuest.Add(new Questions
        {
            CategoryObj = (CategoryQuest)int.Parse(table.Rows[i]["Type"].ToString()),
            Index = int.Parse(table.Rows[i]["idQuestions"].ToString()),
            Value1 = table.Rows[i]["Value1"].ToString(),
            Value2 = table.Rows[i]["Value2"].ToString()
        });
    }
}
```

Рисунок 3.11 – Код считывание вопроса с БД

С помощью конструкции, изображенной на рисунке 3.12, реализуется считывание ответа в опроснике.

```
private WithSelectedValue TaskWithSelectedValue(Questions quest)
{

    var array = quest.Value2.Split(new char[] {'\n','\r'}, StringSplitOptions.RemoveEmptyEntries);
    var listt = new List<_Value>();

    for (int i = 0; i < array.Length; i++)
    {
        if (array[i][0] == '+')
            listt.Add(new _Value { Answer = array[i].Remove(0, 1), Valid = true });
        else
            listt.Add(new _Value { Answer = array[i].Remove(0, 1), Valid = false });
    }
    return new WithSelectedValue
    {
        ID = quest.Index,
        Questions = quest.Value1,
        _Quest = listt
    };
}
```

Рисунок 3.12 – Код считывания ответов

С помощью конструкции, изображенной на рисунке 3.13, реализуется сверка ответа в опроснике.

```
private void button1_Click(object sender, EventArgs e)
{//Задание с выбором ответа

    var _cl = ((WithSelectedValue)dictionary[index].MyClass);

    for (int i = 0; i < checkedListBox1.CheckedItems.Count; i++)
    {
        if (!_cl._Quest.Find(x => x.Answer == checkedListBox1.Items[checkedListBox1.CheckedIndices[i]]).Valid)
        {
            break;
        }
        if(i==checkedListBox1.CheckedItems.Count-1)
            point++;

    }

    CheckPoint();
    SetQuest();
}
```

Рисунок 3.12 – Код сверки ответов

По окончанию опроса вызывается функция, считывающая результат, код изображен на рисунке 3.13.

```
index++;
hod = (100.0 * point / dictionary.Count); // сравнение результата
if (index == dictionary.Count)
{
    WriteResult();
    if ((hod >= 0) && (hod < 15))
    MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \n" +
        "Ваше оборудование впорядке!", user.Name, user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "" +
        "Результат!", MessageBoxButtons.OK, MessageBoxIcon.Information);

    if ((hod >= 15) && (hod < 60))
    MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \n" +
        "Вашему оборудованию рекомендован ремонт!", user.Name, user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "" +
        "Результат!", MessageBoxButtons.OK, MessageBoxIcon.Information);

    if (hod >= 60)
    MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \n" +
        "Вашему оборудованию рекомендована замена!", user.Name, user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "" +
        "Результат!", MessageBoxButtons.OK, MessageBoxIcon.Information);

    Close();
}
```

Рисунок 3.13– Код принятия решений по результату

Далее, полученный результат записывается в СУБД, код изображен на рисунке 3.14.

```
WithSelectedValue quest_WithSelectedValue = (WithSelectedValue)dictionary[index].MyClass;
checkedListBox1.Items.Clear();
textBox1.Text = quest_WithSelectedValue.Questions;
for (int i = 0; i < quest_WithSelectedValue._Quest.Count; i++)
{
    checkedListBox1.Items.Add(quest_WithSelectedValue._Quest[i].Answer);
}
panel2.BringToFront();
break;
```

Рисунок 3.14– Занесение результата опроса в СУБД

4 Разработка документации по использованию информационной системы

4.1 Руководство для пользователя

Для того чтобы начать пользоваться информационной системой пользователю необходимо запустить приложение с помощью файла «Test.exe». После запуска появится диалоговое окно авторизации как на рисунке 4.1.

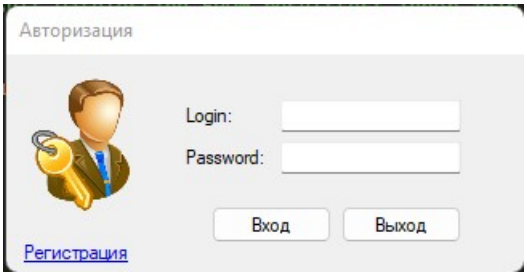

Рисунок 4.1 – Окно авторизации

Пользователю необходимо ввести в соответствующие поля логин и пароль для авторизации. Если же у пользователя отсутствует учетная запись, то ему необходимо зарегистрироваться. Для этого нужно нажать на кнопку «Регистрация» и будет выведено диалоговое окно как на рисунке 4.2.

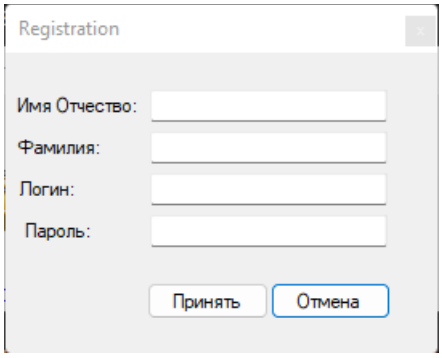

Рисунок 4.2 – Окно регистрации

В данной форме необходимо заполнить все текстовые поля личными данными, а затем нажать на кнопку «Принять».

В случае успешной регистрации пользователь получит диалоговое сообщение как на рисунке 4.3.

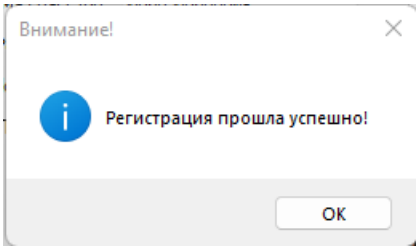

Рисунок 4.3 – Сообщение об успешной регистрации

Пройдя процесс регистрации, пользователь имеет данные для входа в информационную систему. Введя их в окне авторизации и нажав на кнопку «Вход» открывается главное меню приложения как на рисунке 4.4.

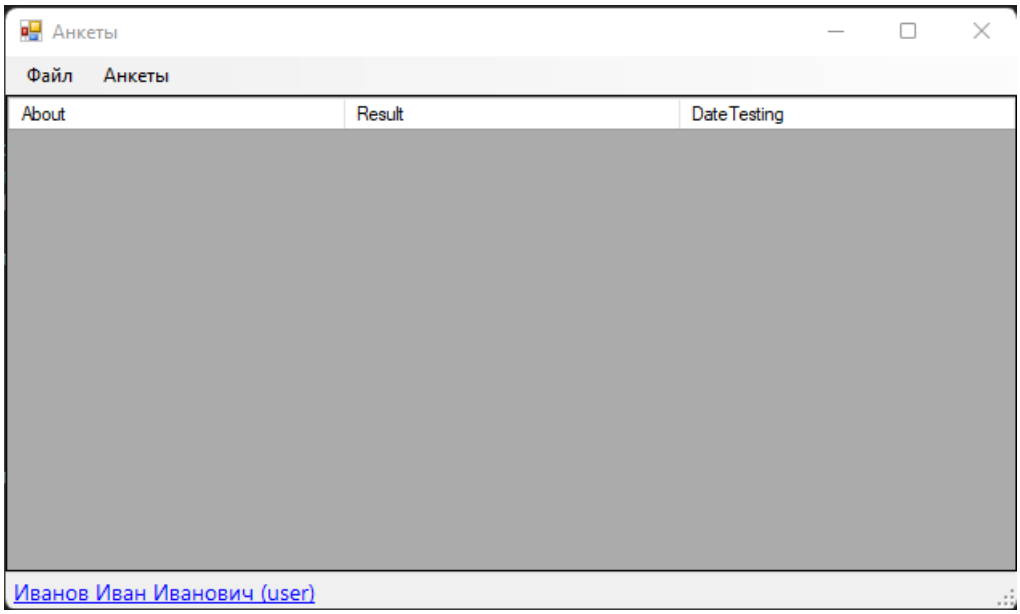

Рисунок 4.4 – Главное меню приложения

Для прохождения опроса необходимо нажать в верхнем меню на кнопку «Анкеты», а после выбрать «Пройти анкету». После нажатия будет осуществлен переход на соответствующую вкладку, где можно выбрать анкету и начнется опрос (рисунок 4.5-6).

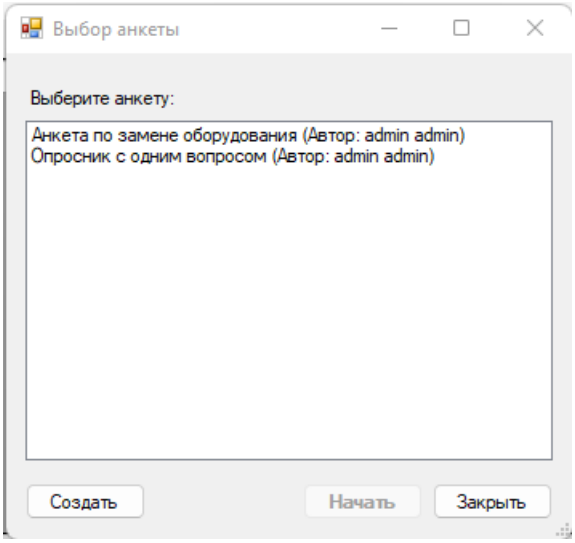

Рисунок 4.5 – Выбор анкеты

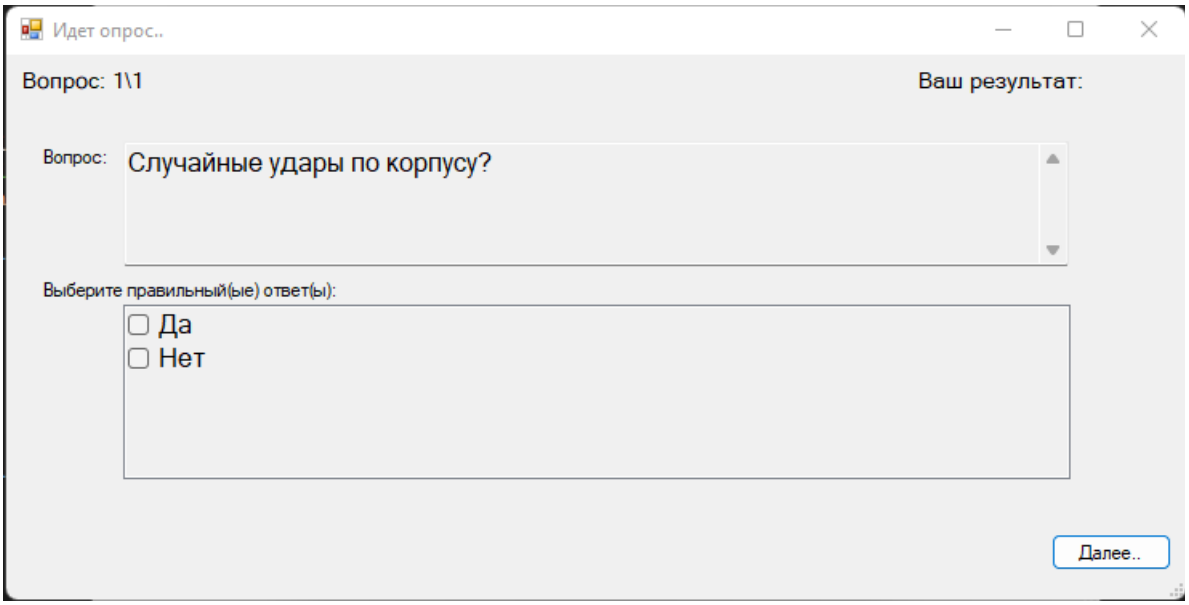

Рисунок 4.6 – Окно с опросом

После прохождения опроса выдается сообщение с решением по замене сетевого оборудования (рисунок 4.7).

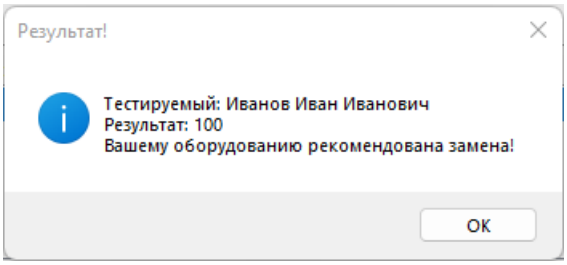

Рисунок 4.7 – Сообщение с результатом опроса

После закрытия сообщения происходит закрытие опроса и возвращение на главную форму, где показан результат прохождения опроса (рисунок 4.8).

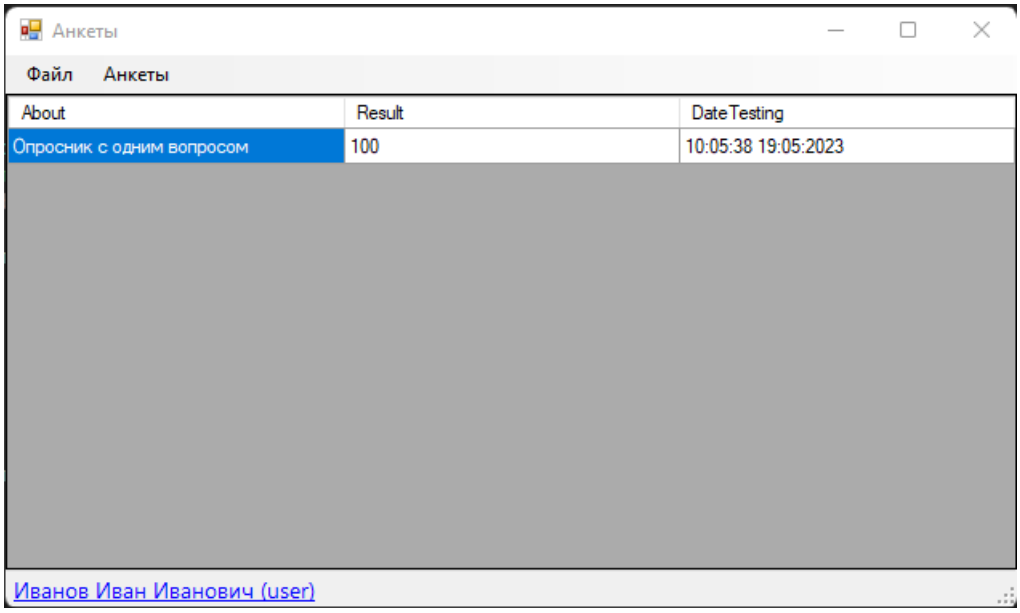

Рисунок 4.8 – Главное меню после прохождения опроса

Чтобы выйти из приложения нужно нажать в верхнем меню кнопку «Файл». После нажатия выбрать «Выход» (рисунок 4.9).

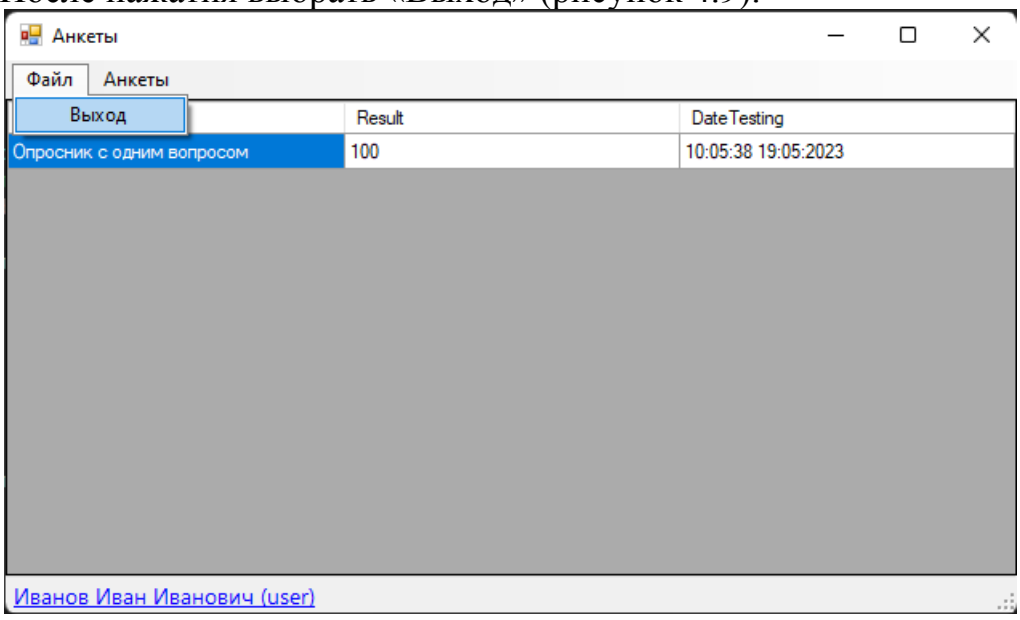

Рисунок 4.9 – Выход из приложения

4.2 Руководство для администратора системы

Основными задачами администратора информационной системы поддержки принятия решений по замене сетевого оборудования организации:

– поддержка целостности базы данных;

– наполнение базы данных новыми анкетами;

– слежение за правильностью заполненных данных и др.

Перед тем как начать управлять информационной системой администратору следует пройти процесс аутентификации (авторизации). Для этого необходимо запустить приложение с помощью файла «Test.exe». После запуска появится диалоговое окно авторизации как на рисунке 4.10.

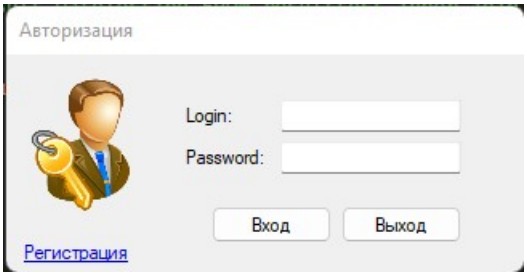

Рисунок 4.10 – Окно авторизации

В данной версии информационной системы имеется лишь одна учетная запись, в которой имеется возможность управления системой. Необходимо ввести следующие данные для входа:

– логин: admin;

– пароль: admin.

Процесс ввода данных представлен на рисунке 4.11.

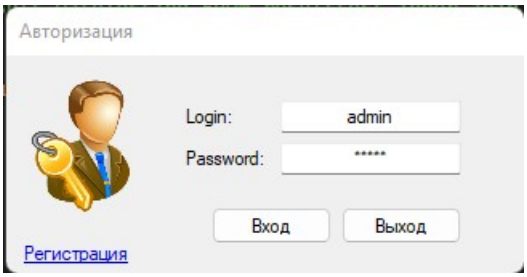

Рисунок 4.11 – Ввод данных

После нажатия на кнопку «Вход» откроется главное меню информационной системы как на рисунке 4.12.

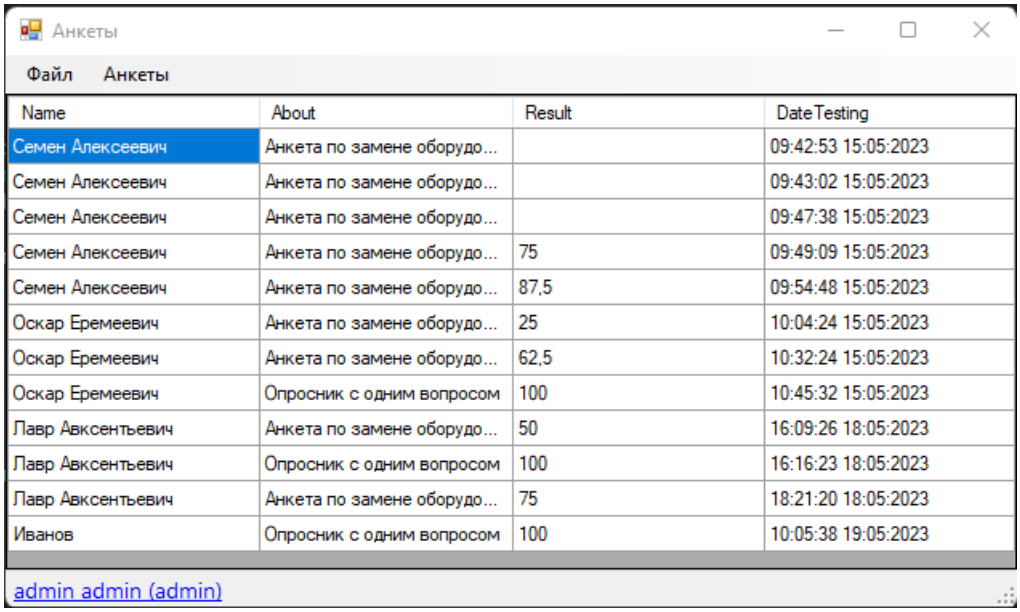

Рисунок 4.12 – Главное меню

Основным отличием главного меню администратора системы от главного меню обычного пользователя является возможность видеть результаты всех опросов.

Одной из функций является возможность создать новые вопросы и анкеты. Для этого необходимо выбрать в верхнем меню кнопку «Анкеты», а после «Создать анкету» (рисунок 4.13).

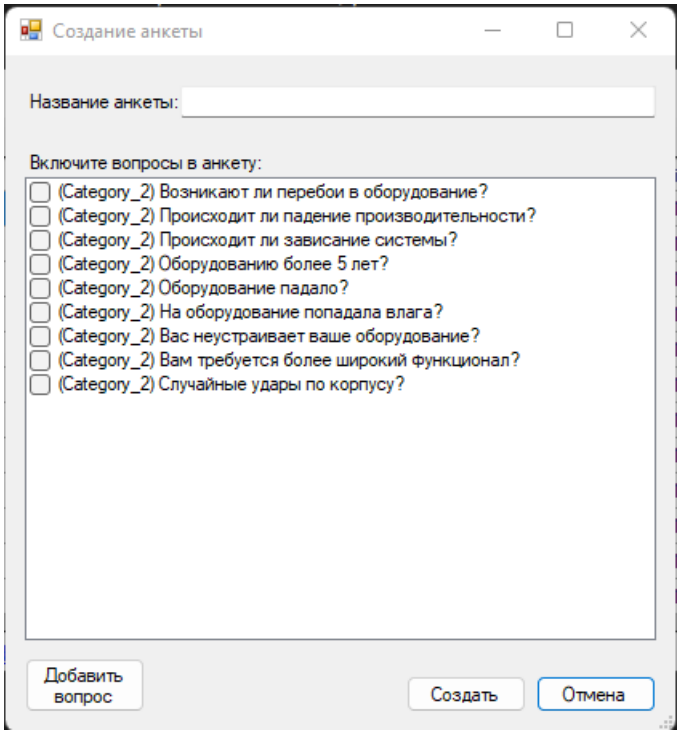

Рисунок 4.13 – Окно создания анкеты

Чтобы добавить новый вопрос необходимо нажать кнопку «Добавить вопрос». И заполнить поля (рисунок 4.14).

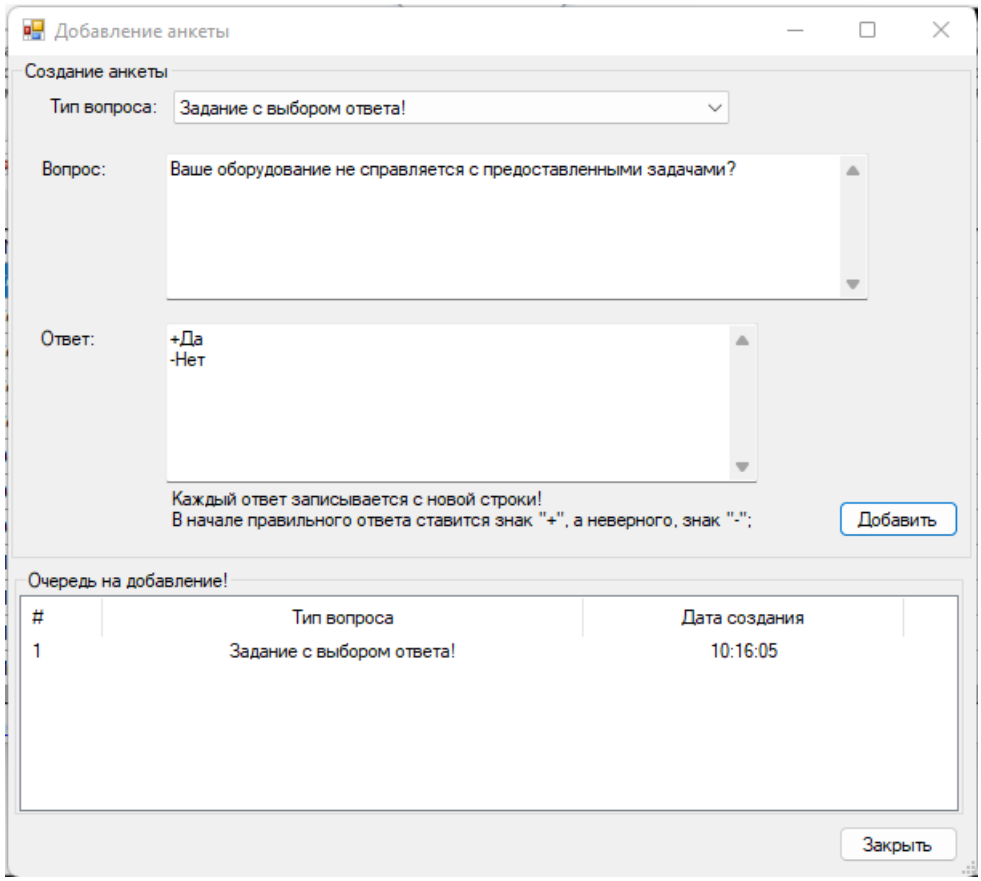

Рисунок 4.14 – Окно добавления вопроса

Чтобы создать новую анкету, нужно вписать имя анкеты и выбрать соответствующие вопросы из списка, представленного ниже (рисунок 4.15).

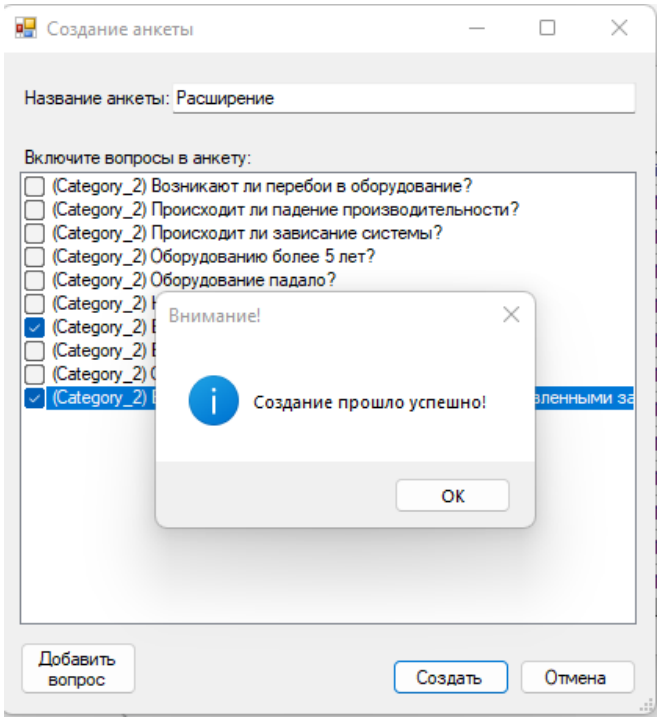

Рисунок 4.15 – Создание анкеты

Чтобы выйти из приложения нужно нажать в верхнем меню кнопку «Файл». После нажатия выбрать «Выход» (рисунок 4.16).

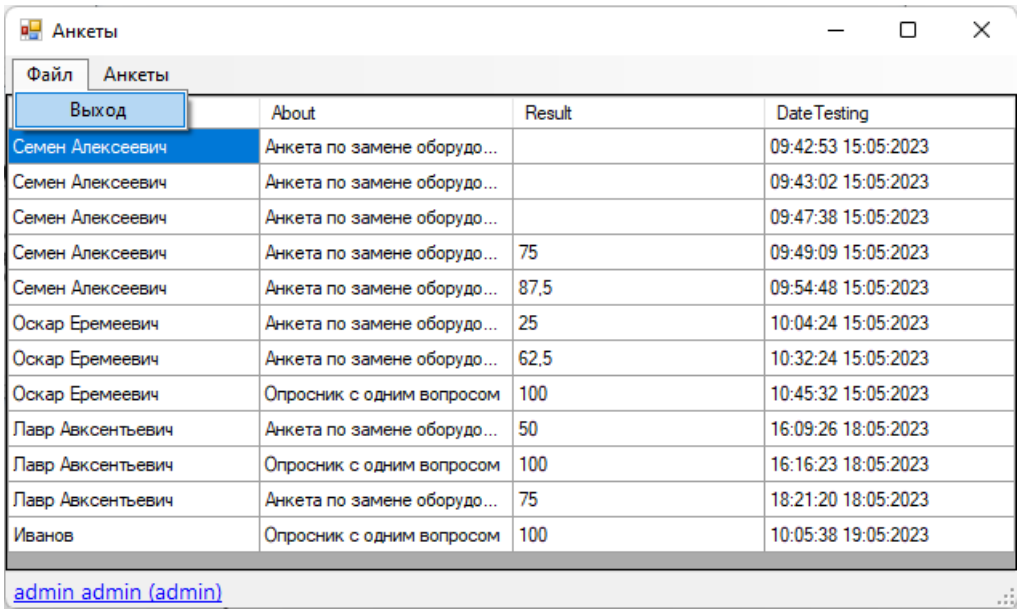

Рисунок 4.16 – Выход из приложения

4.3 Тестирование разработанного приложения

Данная СППР выдает 3 варианта решения по замене сетевого оборудования. Проведем тестирование и посмотрим на результаты.

Для начала требуется запустить приложение через «Test.exe». Далее необходимо авторизоваться в информационной системе и выбрать анкету для прохождения (рисунок 4.17).

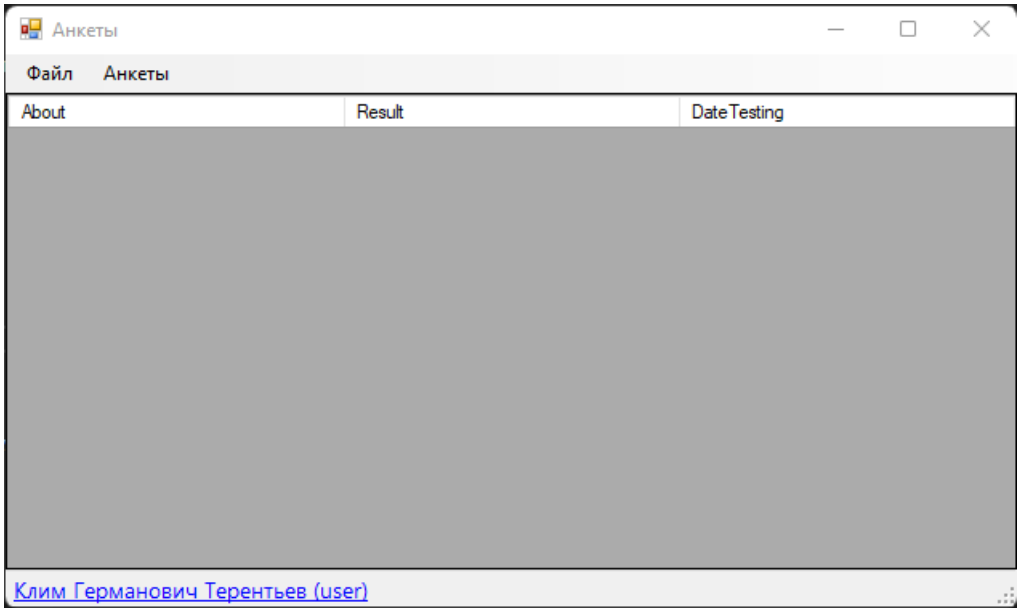

Рисунок 4.17 – Главное меню

После выбора анкеты, начнется опрос. Покажем возможные варианты поведения СППР, которые зависят от набранных баллов (рисунки 4.18-20).

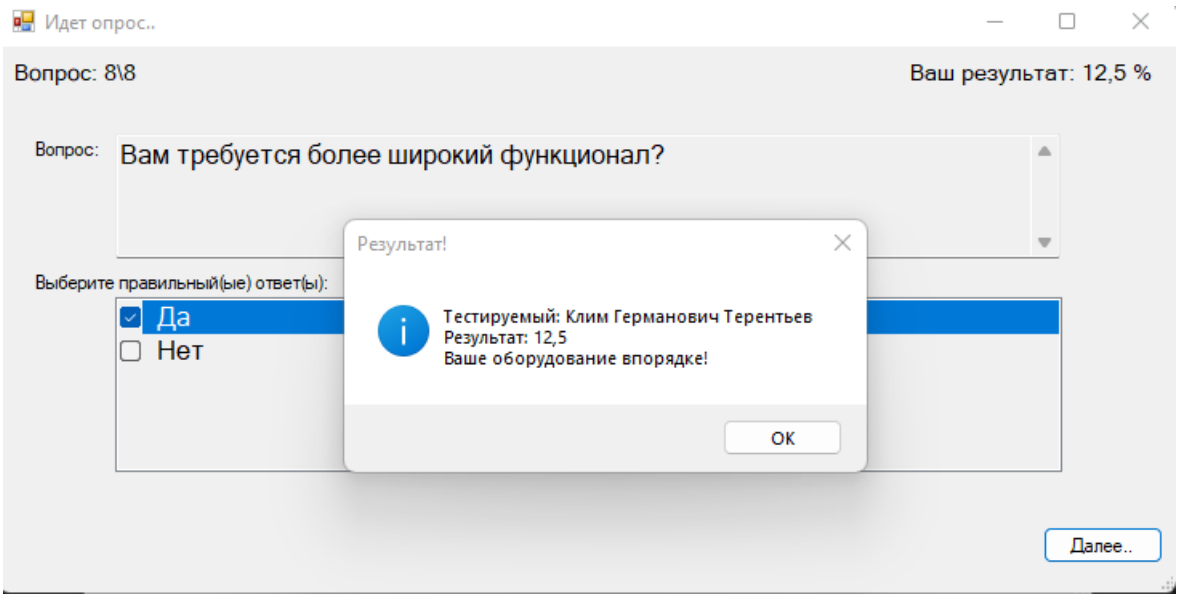

Рисунок 4.18 – Результат первого прохождения опроса

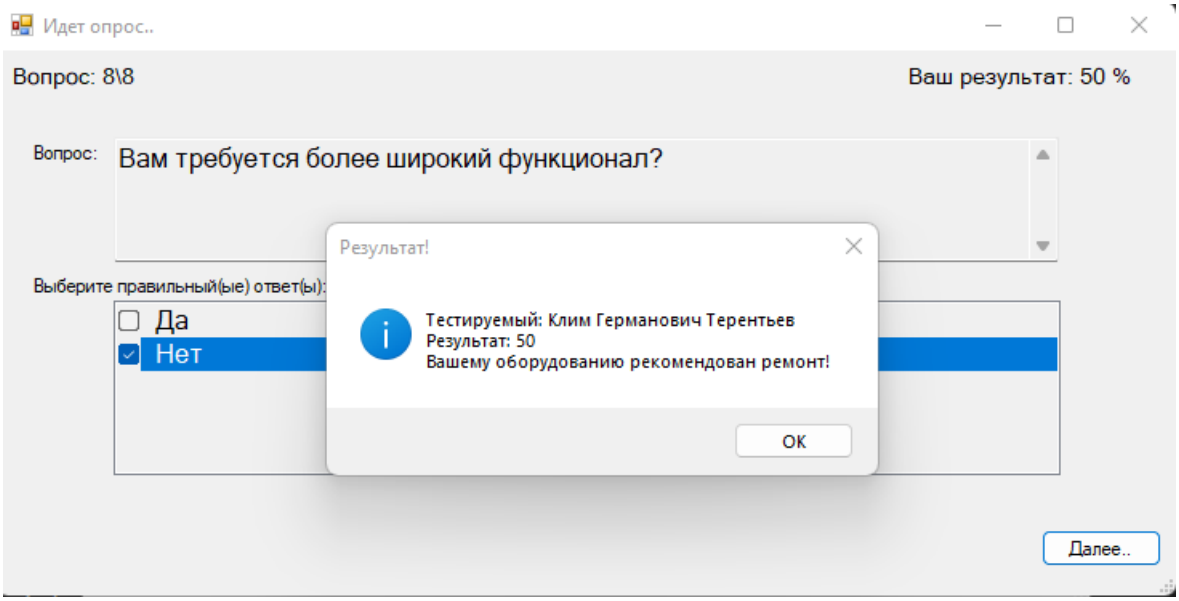

Рисунок 4.19 – Результат второго прохождения опроса

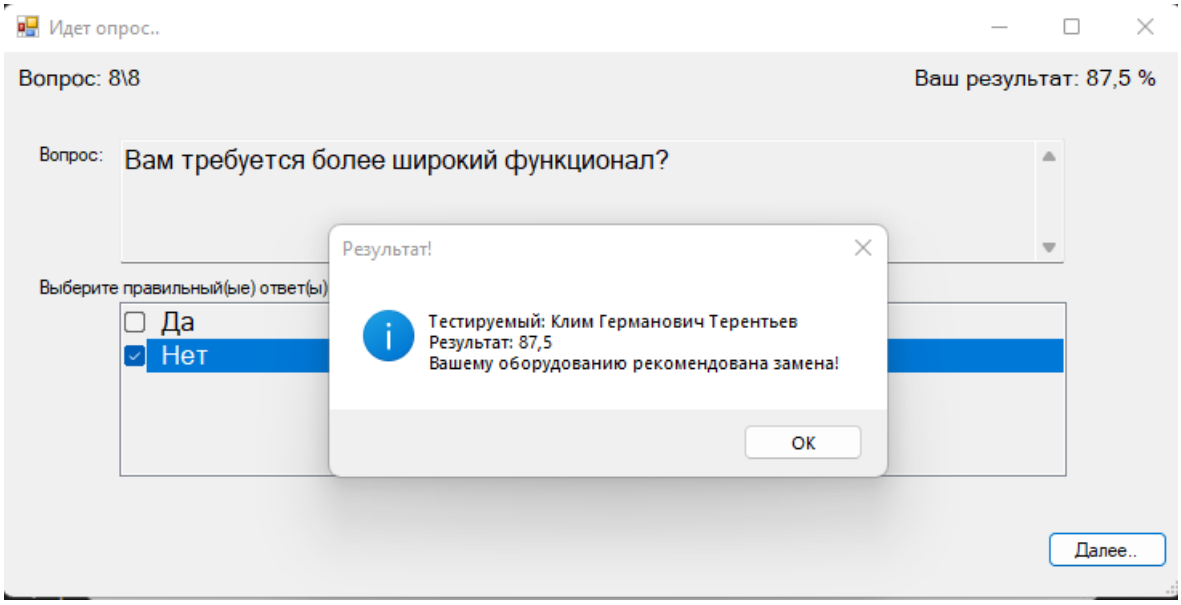

Рисунок 4.20 – Результат третьего прохождения опроса

Как видно, СППР выдает три различных варианта, однако окончательное решение остается за человеком.

Заключение

В ходе прохождения преддипломной практики была достигнута основная цель практики – закрепление и углубление теоретических знаний, полученных студентами в процессе обучения, а также углубленное исследование темы диплома. Было выполнено индивидуальное задание. В данном индивидуальном задании была проведена разработка информационной системы поддержки принятия решений по замене сетевого оборудования.

Осуществлено знакомство с работой предприятия, приобретены навыки профессиональной и организационной деятельности на рабочих местах.

На основании поставленной цели был выполнен ряд задач:

1 Описать профильную организацию ООО «УЦСБ».

Подчеркнем, что это организация, специализирующаяся на создании, модернизации и обслуживании подсистем защиты, а также других услугах в сфере информационной безопасности в организациях и предприятиях

2 Проанализировать предметную область.

Напомним, что СППР – компьютерная автоматизированная система, целью которой является помощь людям, принимающим решение в сложных условиях для полного и объективного анализа предметной деятельности. Это диалоговая система, использующая правила принятия решений и соответствующие модели с базами данных, а также интерактивный компьютерный процесс моделирования.

Для повышения качества проектирования систем поддержки принятия решений прежде всего необходимо внедрение информационных систем. Реализация СППР на основе приложений становится решающим фактором в поддержании СППР как глобального продукта.

Главной особенностью систем поддержки принятия решений становится качественно новый метод организации взаимодействия человека и компьютера.

3 Построить модели информационных процессов.

Залогом успешной реализации проекта по разработке программного обеспечения (ПО) является адекватным образом построенная модель разрабатываемой системы. Использование CASE-средств (Computer Aided Software Engineering) позволяет автоматизировать процессы различных стадий жизненного цикла ПО (ЖЦ ПО). Именно благодаря CASE-средствам разработчики информационных систем (ИС) получили возможность представлять создаваемую систему в наглядной – графической – форме, структурировать процесс разработки модулей системы, облегчить подготовку проектной документации.

4 Обосновать выбор инструментальных средств проектирования и разработки информационной системы.

Исходя из сравнительного анализа функциональный возможностей инструментальных средств был обоснован выбор, так например, за простоту использования, доступность и достаточные возможности, осуществляемые CASE-средствами выбран Ramus.

5 Спроектировать базу данных.

Описана база данных, состоящая из 6 таблиц: «Тестовые анкеты», «Вопросы», «Опросы», «Результаты опросов», «Пользователи». Эти таблицы хранят в себе списки пользователей и их роли, вопросы, анкеты для опросов и результаты опросов.

6 Разработать приложение.

Были описаны используемые методы для реализации приложения.

7 Разработать документацию по использованию информационной системы.

Разработанная документация делится на 2 группы: для пользователя и для администратора системы.

8 Оформить отчет по практике.

Список использованных источников

1 О компании Уральский центр систем безопасности – УЦСБ [Электронный ресурс]. – Режим доступа: https://www.ussc.ru/company/

2 Общероссийская Сеть распространения правовой информации КонсультантПлюс [Электронный ресурс]. Режим доступа: http://www.consultant.ru

3 ООО «УЦСБ», Свердловская область, Екатеринбург (ИНН 6672235068, ОГРН 1076672021194) – реквизиты [Электронный ресурс]. – Режим доступа: https://synapsenet.ru/searchorganization/organization/1076672021194-ooo-ucsb

4 Р. Акофф Искусство решения проблем / Р. Акофф – М.: Книга по Требованию, 2013. – 218 с.

5 Балыбин В.М., Лунев В.С., Муромцев Д.Ю. Принятие проектных решений: Учебное пособие, ч. 1. – Тамбов: Изд-во Тамб. гос. тех. университета, 2013.

6 Алтунин А.Е., Семухин М.В. Модели и алгоритмы принятия решений в нечетких условиях. Тюмень: Изд-во ТГУ, 2010. - 352 с.

7 Саати Т. Принятие решений. Метод анализа иерархий. Пер. с англ. – М.: Радио и связь, 1993.

8 Блюмин С.Л., Шуйкова И.А. Модели и методы принятия решений в условиях неопределенности. Липецк: ЛЭГИ, 2011. - 138 с.

9 Евланов Л.Г. Основы теории принятия решений. М.: АНХ. 2012. – 212 с.

10 Воронина В.В., Мошкин В.С. Разработка приложений для анализа слабоструктурированных информационных ресурсов. Учебное пособие. — Ульяновск: УлГТУ, 2015. — 162 с.

11 Михеев, Ростислав MS SQL Server 2005 для администраторов. / Р. Михеев; БХВ-Петербург - М., 2007. - 534 с. Режим доступа: https://www.labirint.ru/books/268213

12 Морган, С. Проектирование и оптимизация доступа к базам данных Microsoft SQL Server 2005. Учебный курс Microsoft (+ CD-ROM). / С. Морган - Русская Редакция - М., 2008. - 445 с. Режим доступа: https://avidreaders.ru/book/proektirovanie-i-optimizaciya-dostupa-k-bazam.html

13 . Нордфальт, Йенс Ритейл-маркетинг: Практики и исследования; Альпина Диджитал. / Йенс Нордфальт - М., 2015. - 224 с. Режим доступа: https://www.labirint.ru/books/481671

14 Похилько, А. Ф. CASE-технология моделирования процессов с использованием средств BPWin и ERWin: учебное пособие. / А. Ф. Похилько, И. В. Горбачев. – Ульяновск : УлГТУ, 2008. – 120 с.

15 . Грехэм, И. Объектно-ориентированные методы. Принципы и практика. / И. Грехэм. - М.: «Вильямс», 2004. – 1024с.

16 Болодурина, И. П. Проектирование компонентов распределенных информационных систем : учеб. пособие для магистров. / И. П. Болодурина, Т. В. Волкова - М-во образования и науки Рос. Федерации, Федер. гос. бюджет.

образоват. учреждение высш. проф. образования "Оренбург. гос. ун -т". – Оренбург : Университет, 2012. – 216 с. Режим доступа: https://www.iprbookshop.ru/30122

17 Леоненков, А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose [Текст] : учебное пособие. / А. В. Леоненков. - Москва : Интернет-Ун-т Информ. Технологий : БИНОМ. Лаборатория знаний, 2013. - 320 с. : ил. - (Основы информационных технологий). - Библиогр.: с. 317-318. - ISBN 978-5-9556-0043-7. - ISBN 978-5-94774-408-8.

18 Соловьев, И.В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы. / И.В. Соловьев, А.А. Майоров; под ред. В.П. Савиных. - М.: Академический проспект, 2009. - 398 с.

19 Дубенецкий, Б.Я. Проектирование информационных систем. / Б. Я. Дубенецкий. – Л. : ЛЭТИ, 2008 г. – 675 с. Режим доступа: https://etu.ru/sveden/files/RP/7909013/RPD_SpecVoprIsslModKIUS-17z.pdf

20 Бен-Ган, Ицик Microsoft SQL Server 2012. Основы T-SQL. / Ицик Бен-Ган. - М.: Эксмо, 2012. - 167 с. Режим доступа: https://www.labirint.ru/books/471025

21 Федоров, Н.В. Проектирование информационных систем на основе современных CASE-технологий / Н.В. Федоров. - М.: МГИУ, 2008. - 280 с.

22 Калянов Г.Н., Козлинский А.В., Лебедев В.Н. Сравнительный анализ структурных методологий. //СУБД 1997, N 5-6, с. 75-78.

23 Мюллер, Роберт Дж. Проектирование баз данных и UML / Мюллер Роберт Дж.. - М.: ЛОРИ, 2013. - 422 с.

24 Липаев, В.В. Системное проектирование сложных программных средств для информационных систем. / В.В. Липаев. – М. : Синтег, 2009 г. – 156 с.

25 Петкович, Д. Microsoft SQL Server 2016. Руководство для начинающих / Д. Петкович. - М.: БХВ-Петербург, 2017. - 752 с.

26 Выбор эффективного инструмента проектирования программного обеспечения средств [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/vybor-effektivnogo-instrumenta-proektirovaniya-programmnogo-obespecheniya-case-sredstv/viewer

27 Продукты Visio и Project [Электронный ресурс]. – Режим доступа: www.microsoft.com/ru-ru/office/vip/visio.aspx

28 StarUML 2. A sophisticated software modeler [Электронный ресурс]. – Режим доступа: http://staruml.sourceforge.net

29 IBM Rational software [Электронный ресурс]. – Режим доступа: http://www.ibm.com/rationa

30 CASE-технологии. Современные методы и средства проектирования информационных систем [Электронный ресурс]. – Режим доступа: http://citforum.ru/database/case/

31 Сравнение современных СУБД [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/sravnenie-sovremennyh-subd/viewer

Приложение А
(обязательное)

Листинг программного кода

Form.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
   public partial class Form1 : Form
   {
      User user = new User();
      public Form1()
      {
         InitializeComponent();
         AuthForm auth = new AuthForm(user);
         auth.ShowDialog();
         toolStripStatusLabel1.Text = string.Format("{0} {1} ({2})", user.Name, user.SurName, user.UserRole);
         FillTable();
      }

      private void FillTable()
      {
         dataGridView1.DataSource = null;
         SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
         //SqlDataAdapter adapter = new SqlDataAdapter(string.Format("SELECT About,Result,DateTesting FROM resulttest inner
join tests on resulttest.idTest=tests.idTests inner join users on resulttest.idUser = users.idUsers where idUsers={0}", user.ID), connect);
         if (toolStripStatusLabel1.Text == string.Format("admin admin (admin)"))
         //Вывод для админа
         { SqlDataAdapter adapter = new SqlDataAdapter(string.Format("SELECT Name,About,Result,DateTesting FROM resulttest
inner join tests on resulttest.idTest=tests.idTests inner join users on resulttest.idUser = users.idUsers"), connect);
            DataTable table = new DataTable();
            adapter.Fill(table);

            dataGridView1.DataSource = table.DefaultView;
         }
         else
         //Вывод для пользователя
         { SqlDataAdapter adapter = new SqlDataAdapter(string.Format("SELECT About,Result,DateTesting FROM resulttest inner
join tests on resulttest.idTest=tests.idTests inner join users on resulttest.idUser = users.idUsers where idUsers={0}", user.ID), connect);
            DataTable table = new DataTable();
            adapter.Fill(table);

            dataGridView1.DataSource = table.DefaultView;
         }
         //DataTable table = new DataTable();
        // adapter.Fill(table);

         // dataGridView1.DataSource = table.DefaultView;
         // dataGridView1.Columns["idUsers"].Visible = false;
      }

      private void создатьТестToolStripMenuItem_Click(object sender, EventArgs e)
      {
         CreateTest createTest = new CreateTest(user);
         createTest.ShowDialog();
```

```
        }

        private void пройтиТестToolStripMenuItem_Click(object sender, EventArgs e)
        {
            SelectTest selectTest = new SelectTest(user);
            selectTest.ShowDialog();
            FillTable();
        }

        private void выходToolStripMenuItem_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

AuthForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
    public partial class AuthForm : Form
    {
        User user;
        public AuthForm(User _user)
        {
            InitializeComponent();
            this.user = _user;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Environment.Exit(0);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            if (!string.IsNullOrWhiteSpace(textBox1.Text) && !string.IsNullOrWhiteSpace(textBox2.Text))
            {
                SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
                SqlDataAdapter adapter = new SqlDataAdapter(string.Format("select Name,SurName,Login,Role,idUsers from Users where
Login = '{0}' and Password = '{1}'",textBox1.Text,textBox2.Text), connect);
                DataTable table = new DataTable();
                adapter.Fill(table);
                if(table.Rows.Count!=0)
                {
                    user.ID = int.Parse(table.Rows[0]["idUsers"].ToString());
                    user.Name = table.Rows[0]["Name"].ToString();
                    user.SurName = table.Rows[0]["SurName"].ToString();
                    user.Login = table.Rows[0]["Login"].ToString();
                    user.UserRole = (Role)int.Parse(table.Rows[0]["Role"].ToString());

                    Close();
                }
                else
                    MessageBox.Show("Логин или пароль неверны!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            else
                MessageBox.Show("Поля логин и пароль не могут быть пустыми!", "Внимание!", MessageBoxButtons.OK, Message-
BoxIcon.Error);
```

```
        }

        private void linkLabel1_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            Registration reg = new Registration();
            reg.ShowDialog();
        }
    }
}
```

TestingForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
    public partial class TestingForm : Form
    {
        List<Questions> listQuest = new List<Questions>();
        Test test;
        User user;
        double point = 0;
        public TestingForm(Test _test,User _user)
        {
            InitializeComponent();
            test = _test;
            user = _user;
            GetFullTest();
            GetQuest();
        }

        private void WriteResult()
        {
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
                SqlCommand command = new SqlCommand(string.Format("insert into resulttest(idUser,idTest,DateTesting,Result) values({0},{1},'{2}','{3}')",user.ID,test.Index, DateTime.Now.ToString("HH:mm:ss dd:MM:yyyy"),(100.0 * point / dictionary.Count).ToString("#.##")), connect);
            connect.Open();
            command.ExecuteNonQuery();
            connect.Close();
        }
        private void GetFullTest()
        {
            string command = "SELECT * FROM testwithquestions inner join tests ON testwithquestions.idTest=tests.idTests inner join questions on testwithquestions.idQuestion = questions.idQuestions where tests.idTests = " + test.Index.ToString();
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
            SqlDataAdapter adapter = new SqlDataAdapter(string.Format("{0}",command), connect);
            DataTable table = new DataTable();
            adapter.Fill(table);

            for (int i = 0; i < table.Rows.Count; i++)
            {
                listQuest.Add(new Questions
                {
                    CategoryObj = (CategoryQuest)int.Parse(table.Rows[i]["Type"].ToString()),
                    Index = int.Parse(table.Rows[i]["idQuestions"].ToString()),
                    Value1 = table.Rows[i]["Value1"].ToString(),
                    Value2 = table.Rows[i]["Value2"].ToString()
                });
            }
        }
```

```
int index = -1;
double hod = 0;
Dictionary<int, ObjQuest> dictionary = new Dictionary<int, ObjQuest>();
class ObjQuest
{
   public CategoryQuest _Category { get; set; }
   public object MyClass { get; set; }
}
private void GetQuest()
{

   for (int i = 0; i < listQuest.Count; i++)
   {
      switch ((int)listQuest[i].CategoryObj)
      {
                              //case 0: dictionary.Add(i, new ObjQuest { _Category = CategoryQuest.Category_1, MyClass =
TaskWithOpenAnswer(listQuest[i]) }); break;
                                 case 1: dictionary.Add(i, new ObjQuest { _Category =CategoryQuest.Category_2, MyClass
=TaskWithSelectedValue(listQuest[i])}); break;
                               // case 2: dictionary.Add(i, new ObjQuest { _Category =CategoryQuest.Category_3,MyClass =
OrderingSequence(listQuest[i])}); break;
                             // case 3: dictionary.Add(i, new ObjQuest { _Category = CategoryQuest.Category_4, MyClass =
RelevantIdentified(listQuest[i]) }); break;
      }
   }

   SetQuest();
}

private void SetQuest()
{
   index++;
   hod = (100.0 * point / dictionary.Count); // сравнение результата
   if (index == dictionary.Count)
   {
      WriteResult();
      if ((hod >= 0) && (hod < 15))
      MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \nВаше оборудование впорядке!", user.Name,
user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "Результат!", MessageBoxButtons.OK,
MessageBoxIcon.Information);

      if ((hod >= 15) && (hod < 60))
         MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \nВашему оборудованию рекомендован
ремонт!", user.Name, user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "Результат!", MessageBoxButtons.OK,
MessageBoxIcon.Information);

      if (hod >= 60)
         MessageBox.Show(string.Format("Тестируемый: {0} {1} \nРезультат: {2} \nВашему оборудованию рекомендована
замена!", user.Name, user.SurName, (100.0 * point / dictionary.Count).ToString("#.##")), "Результат!", MessageBoxButtons.OK,
MessageBoxIcon.Information);

      Close();
   }
   else
   {
      label5.Text = string.Format("Вопрос: {0}\\{1}", index + 1, dictionary.Count);
      Random r = new Random();
      switch ((int)dictionary[index]._Category)
      {
         case 0:
            WithOpenAnswer quest_WithOpenAnswer = (WithOpenAnswer)dictionary[index].MyClass;
            textBox2.Text = quest_WithOpenAnswer.Question;
            panel3.BringToFront();
            break;
         case 1:
            WithSelectedValue quest_WithSelectedValue = (WithSelectedValue)dictionary[index].MyClass;
            checkedListBox1.Items.Clear();
            textBox1.Text = quest_WithSelectedValue.Questions;
            for (int i = 0; i < quest_WithSelectedValue._Quest.Count; i++)
            {
```

```
                checkedListBox1.Items.Add(quest_WithSelectedValue._Quest[i].Answer);
            }
            panel2.BringToFront();
            break;
        case 2:
            Sequence quest_Sequence = (Sequence)dictionary[index].MyClass;
            Column2.Items.Clear();
            dataGridView1.Rows.Clear();
            var resultMix = quest_Sequence.OrdSeq.OrderBy(x => r.Next()).ToArray();
            Column2.Items.AddRange(resultMix);

            for (int i = 0; i < quest_Sequence.OrdSeq.Length; i++)
            {
                dataGridView1.Rows.Add((i + 1).ToString());
            }

            panel4.BringToFront();
            break;

        case 3:
            dataGridView2.Rows.Clear();
            RelevantId quest_RelevantId = (RelevantId)dictionary[index].MyClass;
            Column3.Items.Clear();
            Column4.Items.Clear();
            var resultMix2 = quest_RelevantId.Listfirst.OrderBy(x => r.Next()).ToArray();
            var resultMix3 = quest_RelevantId.ListSecond.OrderBy(x => r.Next()).ToArray();
            dataGridView2.Rows.Add();
            dataGridView2.Rows.AddCopies(0, resultMix2.Length - 1);
            Column3.Items.AddRange(resultMix2);
            Column4.Items.AddRange(resultMix3);
            panel5.BringToFront();
            break;
    }
  }
}
/*
/// <summary>
/// Задание с открытым ответом
/// </summary>
private WithOpenAnswer TaskWithOpenAnswer(Questions quest)
{
  return new WithOpenAnswer
  {
    ID = quest.Index,
    Answer = quest.Value2,
    Question = quest.Value1
  };
}*/
/// <summary>
/// Задание с выбором ответа
/// </summary>
private WithSelectedValue TaskWithSelectedValue(Questions quest)
{

  var array = quest.Value2.Split(new char[] {'\n','\r'}, StringSplitOptions.RemoveEmptyEntries);
  var listt = new List<_Value>();

  for (int i = 0; i < array.Length; i++)
  {
    if (array[i][0] == '+')
      listt.Add(new _Value { Answer = array[i].Remove(0, 1), Valid = true });
    else
      listt.Add(new _Value { Answer = array[i].Remove(0, 1), Valid = false });
  }
  return new WithSelectedValue
  {
    ID = quest.Index,
    Questions = quest.Value1,
    _Quest = listt
  };
```

```
}
/*
/// <summary>
/// Задание на упорядочивание последовательности
/// </summary>
private Sequence OrderingSequence(Questions quest)
{
   var array = quest.Value1.Split(new char[] { '\n', '\r' },StringSplitOptions.RemoveEmptyEntries).ToArray();

   return new Sequence
   {
      OrdSeq = array
   };
}


/// <summary>
/// Задание на установление соответствия
/// </summary>
/// <param name="quest"></param>
private RelevantId RelevantIdentified(Questions quest)
{
   List<string> list1 = new List<string>();
   List<string> list2 = new List<string>();

   var array1 = quest.Value1.Split(new char[] { '\n', '\r' },StringSplitOptions.RemoveEmptyEntries);

   for (int i = 0; i < array1.Length; i++)
   {
      var a = array1[i].Split(new char[] { '=' }, StringSplitOptions.RemoveEmptyEntries);
      list1.Add(a[0]);
      list2.Add(a[1]);
   }

   return new RelevantId
   {
      ID = quest.Index,
      Listfirst = list1,
      ListSecond = list2
   };
}*/

private void button4_Click(object sender, EventArgs e)
{
   // Задание на установление соответствия
   var _cl = ((RelevantId)dictionary[index].MyClass);

   for (int i = 0; i < dataGridView2.Rows.Count; i++)
   {
      if (dataGridView2.Rows[i].Cells[0].Value != null && dataGridView2.Rows[i].Cells[1].Value != null)
      {
         var first = dataGridView2.Rows[i].Cells[0].Value.ToString();
         var second = dataGridView2.Rows[i].Cells[1].Value.ToString();

         var res = _cl.Listfirst.IndexOf(first);
         if (!(_cl.ListSecond[res] == second))
            break;
         if (i == dataGridView2.Rows.Count - 1)
            point++;
      }
      else
         break;
   }

   CheckPoint();
   SetQuest();
}

private void button3_Click(object sender, EventArgs e)
{//Задание на упорядочивание последовательности
```

```
        var _cl = ((Sequence)dictionary[index].MyClass);
        for (int i = 0; i < dataGridView1.Rows.Count; i++)
        {
            if (!(dataGridView1.Rows[i].Cells[1].Value == _cl.OrdSeq[i]))
                break;
            if (i == dataGridView1.Rows.Count - 1)
                point++;
        }
        CheckPoint();
        SetQuest();
    }
    private void button1_Click(object sender, EventArgs e)
    {//Задание с выбором ответа

        var _cl = ((WithSelectedValue)dictionary[index].MyClass);

        for (int i = 0; i < checkedListBox1.CheckedItems.Count; i++)
        {
            if (!_cl._Quest.Find(x => x.Answer == checkedListBox1.Items[checkedListBox1.CheckedIndices[i]]).Valid)
            {
                break;
            }
            if(i==checkedListBox1.CheckedItems.Count-1)
                point++;

        }

        CheckPoint();
        SetQuest();
    }

    private void button2_Click(object sender, EventArgs e)
    {// Задание с открытым ответом

        if (textBox3.Text.ToLower() == ((WithOpenAnswer)dictionary[index].MyClass).Answer.ToLower())
            point++;

        CheckPoint();
        SetQuest();
    }
    private void CheckPoint()
    {
        label6.Text = string.Format("Ваш результат: {0} %", (100.0 * point / dictionary.Count).ToString("#.##"));
    }
}
class WithOpenAnswer
{
    public int ID { get; set; }
    public string Question { get; set; }
    public string Answer { get; set; }
}
class WithSelectedValue
{
    public int ID { get; set; }
    public string Questions { get; set; }
    public List<_Value> _Quest { get; set; }

}
class _Value
{
    public string Answer { get; set; }
    public bool Valid { get; set; }
}
class Sequence
{
    public string[] OrdSeq { get; set; }
}
class RelevantId
{
```

```
        public int ID { get; set; }
        public List<string> Listfirst { get; set; }
        public List<string> ListSecond { get; set; }
    }
}
```

Select.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
    public partial class SelectTest : Form
    {
        User user;
        List<Test> testsList = new List<Test>();
        public SelectTest(User _user)
        {
            InitializeComponent();
            user = _user;
            GetTests();
        }
        private void GetTests()
        {
            testsList.Clear();
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
            SqlDataAdapter adapter = new SqlDataAdapter("SELECT idTests,About,Name,SurName FROM tests, users where user-
s.idUsers = tests.idUser", connect);
            DataTable table = new DataTable();
            adapter.Fill(table);
            for (int i = 0; i < table.Rows.Count; i++)
            {
                testsList.Add(new Test
                {
                    About = table.Rows[i]["About"].ToString(),
                    Name = table.Rows[i]["Name"].ToString(),
                    SurName = table.Rows[i]["SurName"].ToString(),
                    Index = int.Parse(table.Rows[i]["idTests"].ToString())
                });
            }
            AddTestsInListView();
        }

        private void AddTestsInListView()
        {
            listBox1.Items.Clear();
            for (int i = 0; i < testsList.Count; i++)
            {
                listBox1.Items.Add(string.Format("{0} (Автор: {1} {2})",testsList[i].About,testsList[i].Name,testsList[i].SurName));
            }
        }
        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            CreateTest createTest = new CreateTest(user);
            createTest.ShowDialog();
```

```
            GetTests();
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            button2.Enabled = true;
        }

        private void button2_Click(object sender, EventArgs e)
        {
            TestingForm formTest = new TestingForm(testsList[listBox1.SelectedIndex],user);
            formTest.ShowDialog();
            Close();
        }
    }
}
```

Registration.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
    public partial class Registration : Form
    {
        public Registration()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
            SqlCommand command = new SqlCommand(string.Format("insert into users(Name,SurName,Login,Password,Role)
values('{0}','{1}','{2}','{3}',1)",textBox2.Text,textBox1.Text,textBox3.Text,textBox4.Text), connect);
            try
            {
                connect.Open();
                command.ExecuteNonQuery();
                connect.Close();
                MessageBox.Show("Регистрация прошла успешно!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Informa-
tion);
            }
            catch {
                MessageBox.Show("При регистрации возникла ошибка!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Er-
ror);
            }
            Close();
        }
    }
}
```

CreateTest.cs

```
using System;
using System.Collections.Generic;
```

```
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
   public partial class CreateTest : Form
   {
      List<Questions> list = new List<Questions>();
      User user;
      public CreateTest(User _user)
      {
         InitializeComponent();
         this.user = _user;

         if (user.UserRole == Role.user)
            button3.Enabled = false;
         ReadQuestions();
      }
      private void ReadQuestions()
      {
         checkedListBox1.Items.Clear();
         list.Clear();
         SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
         SqlDataAdapter adapter = new SqlDataAdapter("select * from questions", connect);
         DataTable table = new DataTable();
         adapter.Fill(table);

         for (int i = 0; i < table.Rows.Count; i++)
         {
            list.Add(
               new Questions
               {
                  CategoryObj = (CategoryQuest)int.Parse(table.Rows[i]["Type"].ToString()),
                  Index = int.Parse(table.Rows[i]["idQuestions"].ToString()),
                  Value1 = table.Rows[i]["Value1"].ToString(),
                  Value2 = table.Rows[i]["Value1"].ToString()
               });
         }
         list.OrderBy(x => x.CategoryObj);

         foreach (var item in list)
         {
            checkedListBox1.Items.Add(string.Format("({0}) {1}",item.CategoryObj,item.Value1));
         }

      }

      private void button3_Click(object sender, EventArgs e)
      {
         AddTests addQuestions = new AddTests();
         addQuestions.ShowDialog();

         ReadQuestions();
      }

      private void button2_Click(object sender, EventArgs e)
      {
         try
         {
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
            SqlCommand command = new SqlCommand(string.Format("insert into tests(About,idUser) values('{0}',{1}); SELECT
CAST(scope_identity() AS int)", textBox1.Text, user.ID), connect);
            connect.Open();
            var idTest = (int)command.ExecuteScalar();
```

```
            connect.Close();

            List<string> values = new List<string>();

            for (int i = 0; i < checkedListBox1.CheckedItems.Count; i++)
            {
               values.Add(string.Format("({0},{1})", list[checkedListBox1.CheckedIndices[i]].Index, idTest));
            }

            var result = string.Join(",", values);

            command = new SqlCommand(string.Format("insert into testwithquestions(idQuestion,idTest) values {0}", result), connect);
            connect.Open();
            command.ExecuteNonQuery();
            connect.Close();
            MessageBox.Show("Создание прошло успешно!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Information);
            Close();
         }
         catch
         {
            MessageBox.Show("При создании возникла ошибка!", "Внимание!", MessageBoxButtons.OK, MessageBoxIcon.Error);
         }

      }
   }
}
```

AddTest.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Tests
{
   public partial class AddTests : Form
   {
      public AddTests()
      {
         InitializeComponent();
      }

      private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
      {
         //Задание с открытым ответом!
         //Задание с выбором ответа!
         //Задание на упорядочивание последовательности!
         //Задание на устоновление соответствия!
         switch(comboBox1.SelectedIndex)
         {
            case 0: panelType1.BringToFront(); break;
            case 1: panelType2.BringToFront(); break;
            case 2: panelType3.BringToFront(); break;
            case 3: panelType4.BringToFront(); break;
            default: break;
         }
      }

      int index = 1;
      private void AddInListView()
      {
         ListViewItem listViewItem1 = new ListViewItem(new string[] { index.ToString(), comboBox1.Text, DateTime.Now.ToLong-
TimeString() });
```

```
            listView1.Items.Add(listViewItem1);
            index++;
        }

        private bool WriteDataInDB(string commandText)
        {
            SqlConnection connect = new SqlConnection(Properties.Settings.Default.connectionString);
            SqlCommand command = new SqlCommand(commandText, connect);
            try
            {
                connect.Open();
                command.ExecuteNonQuery();
                connect.Close();
                return true;
            }
            catch {
                return false;
            }
        }

        private void button4_Click(object sender, EventArgs e)
        {
            //Задание на установление соответствия!
            var result = WriteDataInDB(string.Format("insert into questions(Type,Value1)
values({0},'{1}')",comboBox1.SelectedIndex,textBox5.Text));

            if(result)
                AddInListView();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            //Задание на упорядочивание последовательности!
            var result = WriteDataInDB(string.Format("insert into questions(Type,Value1) values({0},'{1}')", comboBox1.SelectedIndex,
textBox6.Text));

            if (result)
                AddInListView();
        }

        private void button1_Click(object sender, EventArgs e)
        {
          // Задание с открытым ответом!
            var result = WriteDataInDB(string.Format("insert into questions(Type,Value1,Value2) values({0},'{1}','{2}')", comboBox1.Se-
lectedIndex, textBox1.Text,textBox2.Text));

            if (result)
                AddInListView();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            //Задание с выбором ответа!
            var result = WriteDataInDB(string.Format("insert into questions(Type,Value1,Value2) values({0},'{1}','{2}')", comboBox1.Se-
lectedIndex, textBox4.Text, textBox3.Text));

            if (result)
                AddInListView();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            Close();
        }
    }
}
```

Дневник по преддипломной практике

Вид, тип практики преддипломная

Обучающийся Чернышов Владимир Константинович
(Фамилия, Имя, Отчество)

Курс 4
Факультет (филиал, институт) Институт математики и информационных технологий
Форма обучения очная
Направление подготовки (специальность) 09.03.02 Информационные системы и технологии
Место прохождения практики ООО «Уральский центр систем безопасности»
(наименование профильной организации)

Срок прохождения практики: с 10.04.2023 по 20.05.2023

Дата (период)	Содержание выполненных работ
10.04	Вводный инструктаж по технике безопасности на рабочем месте
	Ознакомление с историей развития, структурой и задачами предприятия ООО «УЦСБ»
11.04-14.04	Анализ предметной области
17.04-21.04	Исследование моделей информационных процессов информационной системы
24.04 - 12.05	Проектирование базы данных
	Разработка приложения
15.05 - 18.05	Завершение разработки документации, разработки руководства
19.05	Оформление отчета, разработка презентации

Руководитель практики от организации _____ А. В. Манжосов
подпись, печать

Отзывы руководителя практики

от предприятия (организации) о работе студента

За время прохождения преддипломной практики В. К. Чернышов проявил грамотность, а также способность собирать и анализировать информацию, умение использовать полученные знания для решения практических задач.

Успешно применял полученные в университете теоретические знания в области информационных технологий, закрепляя и развивая их в процессе преддипломной практики.

За время работы практикант ознакомился с деятельностью организации и выполнил указанные в индивидуальном задании виды работ в полном объеме и в срок.

Ко всем заданиям относился ответственно, качественно выполнял поставленные задачи.

Оценка за практику _____.

Руководитель практики от организации _____ А. В. Манжосов

подпись, печать