**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М.Ф. Решетнева»**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Институт (факультет) | | | Институт инженерной экономики | |
| Направление | | 09.03.03 «Прикладная информатика» | | |
| Направленность (профиль) | | | | «Цифровые инновации в управлении |
| предприятиями» | | | | |
| Кафедра | Информационных экономических систем | | | |

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

Вид ВКР: бакалаврская работа

|  |
| --- |
| **РАЗРАБОТКА ПРОГРАММНОГО ВИРТУАЛЬНОГО ПОМОЩНИКА** |
| **ПО РАБОТЕ С КЛИЕНТАМИ И ЗАКАЗАМИ** |
| **СЕРВИСНОГО ЦЕНТРА** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Обучающийся |  | / | В. С. Мацук | / |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Руководитель |  | / | А. А. Павленко | / |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ответственный за нормоконтроль |  | / | Н. Ю. Юферова | / |

Допускается к защите

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заведующий кафедрой |  | / | М. А. Масюк | / |

«20» июня 2025 г.

Красноярск 2025

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Сибирский государственный университет науки и технологий**

**имени академика М. Ф. Решетнева»**

|  |
| --- |
| Институт инженерной экономики |
| институт |
| Кафедра информационных экономических систем |
| кафедра |

|  |  |  |  |
| --- | --- | --- | --- |
| УТВЕРЖДАЮ | | | |
| Заведующий кафедрой | | | |
|  |  | | М. А. Масюк |
|  | | «24» декабря 2024 г. | |

**ЗАДАНИЕ**

**НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ**

**в форме бакалаврской работы**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обучающийся | | Мацук Владислав Сергеевич | | | | | | | | | | | |
| Группа | БПЦ21-01 | | | Направление (специальность) | | | | 09.03.03 | | | | | |
| «Прикладная информатика» | | | | | | | | | | | | | |
| Тема выпускной квалификационной работы | | | | | | | Разработка программного | | | | | | |
| виртуального помощника по работе с клиентами и заказами сервисного | | | | | | | | | | | | | |
| центра | | | | | | | | | | | | | |
| Утверждена приказом по университету от | | | | | | 17.03 | | 2025 г. № | | | 744д | | |
| Руководитель ВКР | | | Павленко А. А. старший преподаватель кафедры ИЭС | | | | | | | | | | |
|  | | | | |  | | | | | | | | |
| Исходные данные для ВКР | | | | | Информация о деятельности сервисного | | | | | | | | |
| центра ООО «Термо-Мастер сервис» | | | | | | | | | | | | | |
| Перечень разделов ВКР | | | | | | | | | | | | | |
| 1. Анализ предметной области. | | | | | | | | | | | | | |
| 1. Проектирование системы. | | | | | | | | | | | | | |
| 1. Разработка системы. | | | | | | | | | | | | | |
| Перечень графического материала (с указанием обязательных чертежей | | | | | | | | | | | | | |
| при необходимости): | | | | презентация, раздаточный материал | | | | | | | | | |
| 1. Срок сдачи студентом первого варианта ВКР – | | | | | | | | | | «06» июня 2025 г | | | |
| 1. Срок сдачи студентом окончательного варианта ВКР | | | | | | | | | | «20» июня 2025 г | | | |
| Руководитель ВКР | | | | | | | | / | | А. А. Павленко | | / | |
| Задание принял к исполнению | | | | | | | | / | | В. С. Мацук | | | / |
|  | | | | | | | | | «24 декабря» 2024 г. | | | | |

**СОДЕРЖАНИЕ**

[Введение 4](#_Toc201197302)

[1 Анализ Предметной Области 6](#_Toc201197303)

[1.1 Анализ отрасли 6](#_Toc201197304)

[1.2 Анализ организации 7](#_Toc201197305)

[1.3 Анализ готовых решений 10](#_Toc201197306)

[1.4 Обоснование необходимости разработки 14](#_Toc201197307)

[2 Проектирование системы 16](#_Toc201197308)

[2.1 Требования к системе 16](#_Toc201197309)

[2.2 Функциональная модель системы 17](#_Toc201197310)

[2.3 Архитектура системы 26](#_Toc201197311)

[2.4 Моделирование данных 28](#_Toc201197312)

[2.5 Моделирование логики работы 30](#_Toc201197313)

[3 Разработка системы 36](#_Toc201197314)

[3.1 Этапы разработки 36](#_Toc201197315)

[3.2 Средства разработки 38](#_Toc201197316)

[3.3 Интерфейс пользователя 39](#_Toc201197317)

[3.6 Серверная часть системы 44](#_Toc201197318)

[3.7 Входные данные и выходные данные 48](#_Toc201197319)

[3.8 Тестирование системы 50](#_Toc201197320)

[3.9 Экономическая эффективность разработки системы 51](#_Toc201197321)

[3.10 Перспективы развития системы 53](#_Toc201197322)

[Заключение 56](#_Toc201197323)

[Список использованных источников 57](#_Toc201197324)

ВВЕДЕНИЕ

В современном мире сфера сервисного обслуживания занимает важное место в системе взаимодействия между компаниями и их клиентами. Сервисные центры становятся неотъемлемой частью повседневной жизни, поскольку обеспечивают техническую поддержку, обслуживание и сопровождение различного оборудования и услуг. По мере расширения клиентской базы и увеличения объёма обращений возрастает нагрузка на сотрудников, отвечающих за приём и обработку заявок. В этих условиях особенно актуальным становится стремление к совершенствованию процессов обслуживания, повышению их скорости, доступности и эффективности.

Информационные технологии проникают во все области, и сфера сервисного обслуживания не является исключением. Цифровая трансформация охватывает не только производственные процессы, но и коммуникационные каналы, формируя новые подходы к организации взаимодействия с клиентами. Всё чаще в арсенале инструментов компаний появляются решения, основанные на интеллектуальных алгоритмах, машинном обучении и обработке естественного языка. Эти технологии открывают новые горизонты для повышения качества сервиса и оптимизации работы сотрудников [11].

Одним из таких решений становится использование виртуальных помощников — программных систем, способных в автоматическом режиме обрабатывать запросы, предоставлять справочную информацию, регистрировать обращения и сопровождать клиента на протяжении всего жизненного цикла обращения. Появление таких инструментов является закономерным ответом на возрастающие требования к скорости и точности обслуживания. Они позволяют обеспечить круглосуточную поддержку, снизить нагрузку на персонал и создать более удобный и понятный интерфейс для клиентов.

Развитие интеллектуальных помощников опирается на достижения в области анализа текста, распознавания пользовательских намерений и моделирования диалогов. Такие системы стремятся к приближению к естественному человеческому общению, адаптируются под особенности бизнеса и могут быть интегрированы в существующие каналы коммуникации, включая мессенджеры, веб-интерфейсы и мобильные приложения. Их внедрение становится частью более широкой стратегии цифровизации клиентского сервиса, где акцент делается на оперативность, персонализацию и постоянное присутствие в информационном пространстве клиента.

Таким образом, интеллектуальные программные агенты занимают всё более значимое место в структуре современных сервисных систем, создавая предпосылки для качественных изменений в принципах работы с клиентами. Внедрение таких решений позволяет организациям не только соответствовать текущим требованиям рынка, но и формировать устойчивое конкурентное преимущество, основанное на инновационном подходе к взаимодействию с пользователями [13].

Целью данного проекта является создание программного виртуального помощника, предназначенного для обработки клиентских обращений и заказов в условиях работы сервисного центра. В современных реалиях, когда эффективность и оперативность обслуживания становятся ключевыми факторами конкурентоспособности, внедрение цифровых инструментов взаимодействия с клиентами приобретает особую значимость. Разработка виртуального помощника направлена на повышение скорости обработки обращений, снижение нагрузки на персонал и улучшение общего качества сервиса за счёт автоматизации рутинных операций.

Для достижения поставленной цели необходимо последовательно решить ряд взаимосвязанных задач. В первую очередь требуется провести анализ существующего порядка обработки заказов, выявить основные этапы и особенности текущего взаимодействия между клиентами и сотрудниками сервисного центра. Это позволит сформировать чёткое представление о функциональных требованиях к будущей системе и определить ключевые точки, в которых автоматизация способна принести наибольший эффект.

Следующим этапом является проектирование архитектуры виртуального помощника, включая описание логики его работы, сценариев взаимодействия с пользователями, структуры хранения данных и механизмов обработки информации. Важно предусмотреть гибкость и масштабируемость решения, а также возможность интеграции с другими системами, используемыми в организации.

После этапа проектирования следует непосредственно реализация программного продукта. Разработка виртуального помощника предполагает создание пользовательского интерфейса, реализацию алгоритмов обработки запросов, настройку диалоговых сценариев и обеспечение взаимодействия с внутренними сервисами предприятия. Важно также провести тестирование системы, проверить её на устойчивость и корректность работы в различных условиях.

Завершающим этапом является обобщение полученных результатов, оценка достигнутых показателей, а также формулировка выводов и рекомендаций по дальнейшему развитию и применению системы. Это позволит определить, в какой степени поставленные цели были достигнуты, и наметить возможные направления для будущего совершенствования разработанного решения [14].

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Анализ отрасли

Анализ предметной области представляет собой важнейший этап разработки информационной системы, так как он позволяет глубже понять специфику функционирования организации, для которой создаётся программный продукт, а также выявить ключевые процессы, нуждающиеся в автоматизации. В рамках данного проекта предметной областью выступает деятельность сервисного центра, осуществляющего приём, обработку и выполнение заказов на техническое обслуживание и ремонт оборудования.

Сервисные центры играют роль связующего звена между производителями техники и конечными пользователями. Они обеспечивают техническую поддержку, проводят диагностику неисправностей, выполняют ремонтные работы, предоставляют консультации, а также занимаются оформлением гарантийных и послегарантийных обращений. Учитывая высокую частотность и разнообразие запросов, а также потребность в оперативном реагировании, эффективность работы таких центров напрямую зависит от качества организации взаимодействия с клиентами.

Клиентский поток в сервисных центрах формируется как за счёт первичных обращений, так и повторных заявок, связанных с уточнением информации, отслеживанием статуса заказа или дополнительными вопросами. В условиях увеличения объёма обращений возникает необходимость в чёткой структуре обработки запросов, наличии системы учёта заказов, отслеживании сроков выполнения и оперативном предоставлении информации клиенту. Традиционная схема взаимодействия с использованием телефонной связи или личного приёма требует значительных временных и человеческих ресурсов, что не всегда оправдано с точки зрения эффективности и качества сервиса.

Анализ деятельности показывает, что наиболее уязвимыми звеньями в цепочке обслуживания являются первичная регистрация обращений, предоставление стандартной справочной информации и передача данных между отделами. Именно на этих этапах наиболее целесообразно внедрение автоматизированных инструментов, способных взять на себя обработку повторяющихся и однотипных запросов. Такой подход позволяет разгрузить операторов, сократить время отклика, снизить вероятность ошибок и повысить общую прозрачность процесса обслуживания.

Современные технологии в области обработки естественного языка и построения диалоговых систем позволяют создавать интеллектуальные решения, имитирующие поведение живого консультанта. Такие виртуальные помощники способны вести диалог с клиентом, регистрировать обращения, направлять их в соответствующие отделы, предоставлять информацию о статусе заказа и выполнять множество других операций без участия человека. Их применение открывает широкие возможности для повышения качества обслуживания, улучшения пользовательского опыта и оптимизации внутренних бизнес-процессов [5].

1.2 Анализ организации

Общество с ограниченной ответственностью «Термо-Мастер сервис» — это компания, осуществляющая свою деятельность на территории города Красноярска, по адресу: улица Авиаторов, дом 25А, помещение в подвальном этаже. Организация зарегистрирована 31 января 2017 года и с момента основания стабильно ведёт хозяйственную деятельность в нескольких направлениях, охватывающих как торговлю, так и производственно-ремонтную сферу.

Основной деятельностью предприятия является оптовая торговля прочими машинами и оборудованием. Это направление охватывает широкий спектр поставок различной техники и комплектующих, что требует не только надёжных логистических решений, но и технической экспертизы. В рамках дополнительной специализации компания занимается производством металлических строительных конструкций и их частей, что указывает на наличие производственных мощностей и квалифицированного инженерного персонала. Кроме того, важной составляющей деятельности предприятия является оказание услуг по ремонту различного оборудования — от машин до электронных и электрических устройств, включая электронно-оптическую технику.

Такая многоотраслевая структура работы требует чёткой организации процессов и высокой степени координации между различными подразделениями. Ремонтные работы, проводимые в рамках нескольких ОКВЭД, предполагают взаимодействие с частными и корпоративными заказчиками, техническую диагностику, оформление заказов, отслеживание состояния оборудования и соблюдение сроков исполнения. Всё это формирует высокую нагрузку на сотрудников, особенно в части обработки входящих обращений, консультаций клиентов и административного сопровождения заказов.

Компания зарегистрирована с уставным капиталом в размере 10 000 рублей и имеет двоих совладельцев. Руководство организацией осуществляет директор Макаров Сергей Владимирович. За период своего существования предприятие успело принять участие в одном государственном тендере, а также стало объектом одного исполнительного производства, что является незначительным показателем для организации, работающей в технически сложной отрасли. При этом ООО «Термо-Мастер сервис» не фигурировало в арбитражных делах, что свидетельствует о стабильности его хозяйственных отношений.

Таким образом, деятельность компании охватывает как коммерческую торговлю, так и техническое обслуживание, что делает её примером организации, где актуальны задачи автоматизации клиентского взаимодействия. Рост числа заявок, необходимость поддержания высокого уровня сервиса и стремление к оптимизации внутренних процессов делают внедрение современных цифровых решений, таких как виртуальные помощники, логичным этапом её развития [11].

Организационная структура представлена на рисунке 1.

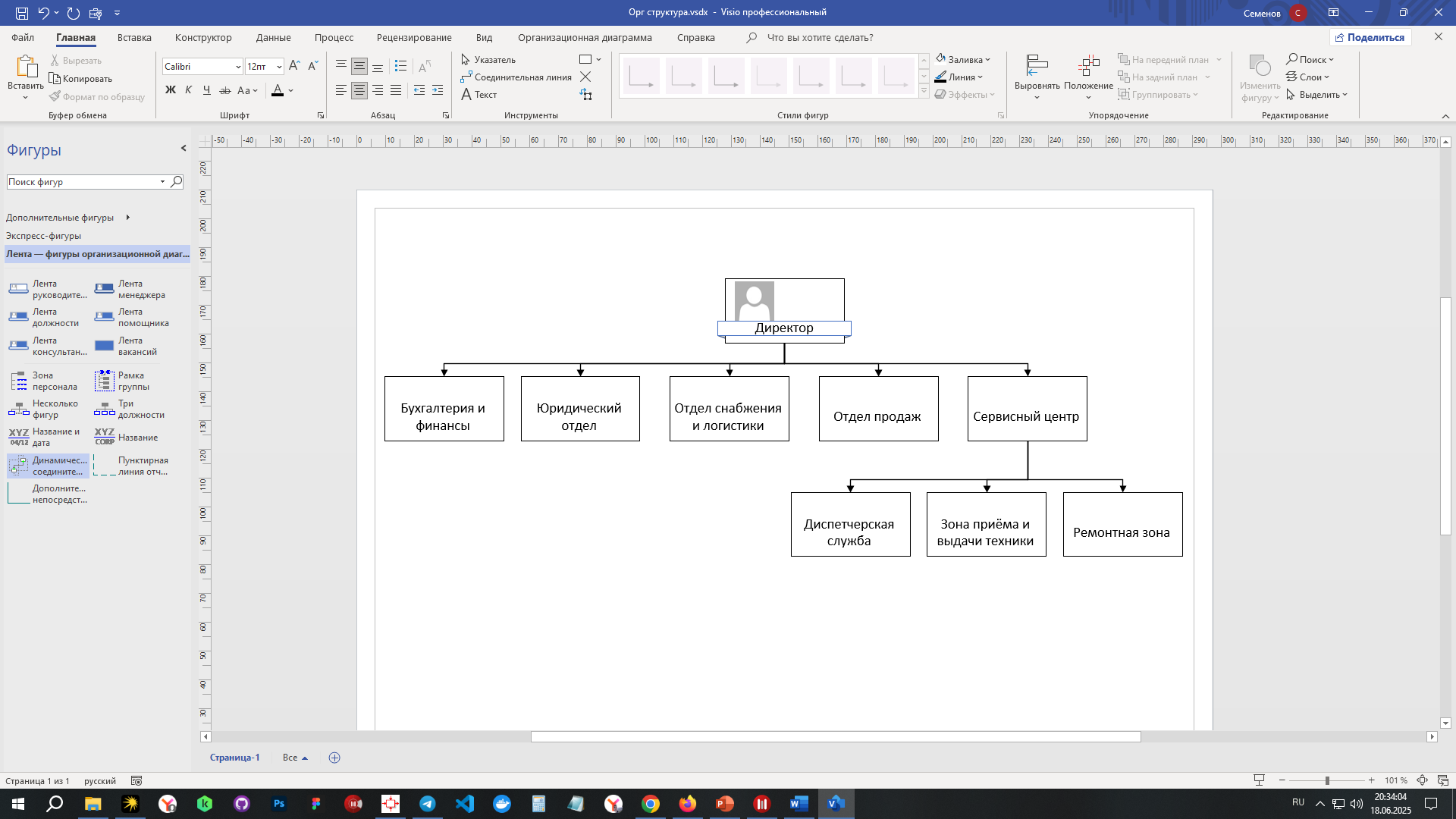


Рисунок 1 – Организационная структура ООО «Термо-Мастер сервис»

Организационная структура ООО «Термо-Мастер сервис» основана на принципах линейно-функционального управления, при котором стратегические решения централизуются на уровне директора, а исполнение задач делегировано специализированным функциональным подразделениям. Такая модель обеспечивает чёткое распределение обязанностей, упрощает контроль исполнения и способствует эффективной координации между отделами. Организация включает в себя как административные, так и производственно-технические звенья, каждое из которых играет свою роль в обеспечении бесперебойной деятельности предприятия.

Высшим звеном в структуре является директор, который осуществляет общее руководство организацией. Он определяет приоритеты развития, контролирует реализацию текущих задач, обеспечивает взаимодействие между подразделениями и принимает ключевые управленческие решения. Директору подчиняются руководители всех основных отделов, охватывающих финансово-экономическую, юридическую, производственную, торговую и сервисную сферы [13].

К числу административных подразделений относятся:

1. Бухгалтерия и финансовый отдел, который отвечает за ведение учёта, расчёты с контрагентами, формирование налоговой и бухгалтерской отчётности, а также за финансовое планирование и анализ.
2. Юридическая служба, выполняющая сопровождение договоров, консультации по правовым вопросам, кадровое делопроизводство и взаимодействие с государственными структурами.
3. Отдел продаж, занимающийся оформлением заказов, взаимодействием с клиентами, подготовкой коммерческих предложений, а также анализом рыночных условий и продвижением продукции компании.

Производственно-логистические процессы курирует:

1. Отдел снабжения и логистики, который обеспечивает поставку необходимого оборудования, материалов и запасных частей, ведёт учёт складских остатков и организует транспортировку.
2. Производственный участок, специализирующийся на изготовлении металлических конструкций и нестандартных деталей. Он взаимодействует с:
   1. Отделом проектирования и разработки, который разрабатывает чертежи и технологическую документацию;
   2. Отделом технического контроля, занимающимся проверкой качества готовой продукции и соблюдением технических стандартов.

Ключевым функциональным элементом в структуре предприятия выступает сервисный центр, который является основным звеном взаимодействия с конечными клиентами. Он имеет собственную внутреннюю структуру, обеспечивающую полный цикл обработки заказов на ремонт и техническое обслуживание оборудования.

Сервисный центр включает в себя:

1. Диспетчерскую службу, принимающую обращения клиентов по различным каналам связи (телефон, электронная почта, мессенджеры). Здесь осуществляется первичная регистрация заявок, консультации по типовым вопросам и распределение задач между ремонтными специалистами.
2. Зону приёма и выдачи техники, где проводится осмотр поступившего оборудования, оформляются документы на приём в ремонт, а после завершения работ техника передаётся клиенту.
3. Ремонтную зону, где работают квалифицированные специалисты, разделённые по направлениям:
   1. Ремонт электрического оборудования;
   2. Ремонт механического и оптического оборудования.
4. Работа сервисного центра требует постоянного взаимодействия с другими отделами компании — с логистикой (по вопросам запчастей), с бухгалтерией (по расчётам с клиентами), с отделом продаж (в случае предложений на замену оборудования), и с юридической службой (при обслуживании по гарантийным обязательствам или оформлении договоров на техническое обслуживание).
5. Такая структура позволяет ООО «Термо-Мастер сервис» эффективно решать задачи, связанные как с оптовой торговлей, так и с производством и техническим обслуживанием. Наличие интегрированного сервисного центра в структуре компании даёт возможность быстро реагировать на потребности клиентов, минимизировать сроки выполнения работ и поддерживать высокий уровень качества обслуживания. В современных условиях, когда возрастает объём входящих обращений и усложняются процессы управления, важным направлением развития такой структуры становится внедрение цифровых инструментов, направленных на автоматизацию взаимодействия с клиентами [4].

С учётом особенностей организационной структуры ООО «Термо-Мастер сервис» и значимой роли сервисного центра в ежедневной деятельности предприятия, разработка и внедрение программного виртуального помощника представляется особенно актуальной и своевременной. Именно сервисный центр является точкой пересечения большинства внутренних процессов и внешних коммуникаций: здесь обрабатываются обращения клиентов, оформляются заявки, осуществляется приём и выдача оборудования, координируется работа технических специалистов и формируется первичное впечатление клиента о компании.

В условиях растущего потока заявок и высокой интенсивности взаимодействия с потребителями, использование интеллектуального программного инструмента, способного автоматизировать первичную обработку запросов, предоставлять справочную информацию, регистрировать обращения и маршрутизировать их соответствующим исполнителям, позволит существенно повысить оперативность и качество обслуживания. Виртуальный помощник станет не просто вспомогательным интерфейсом, а логически встроенным элементом существующей структуры, разгружающим диспетчерскую службу, сокращающим время ожидания отклика и минимизирующим вероятность ошибок, связанных с человеческим фактором.

Таким образом, интеграция виртуального помощника в процессы сервисного центра является обоснованным шагом на пути к цифровизации внутренней инфраструктуры предприятия и повышению его клиенториентированности. Это решение не только отвечает современным тенденциям развития сервисных организаций, но и органично вписывается в текущую модель управления компании, обеспечивая дополнительный уровень эффективности и устойчивости её работы [16].

1.3 Анализ готовых решений

Проведение сравнительного анализа является важным этапом при выборе программного обеспечения для автоматизации бизнес-процессов, особенно в условиях ограниченных ресурсов и высокой ответственности за результат. Такой анализ позволяет систематизировать информацию о возможных вариантах, выявить сильные и слабые стороны каждого решения, сопоставить их с текущими и перспективными задачами организации, а также обосновать выбор наиболее рационального варианта на основе объективных критериев.

Главное преимущество сравнительного анализа — это возможность принимать решения не интуитивно, а опираясь на данные. Вместо того чтобы руководствоваться внешней привлекательностью продукта, рекламой или популярностью на рынке, анализ помогает сфокусироваться на конкретных технических и организационных характеристиках, таких как уровень поддержки искусственного интеллекта, наличие интеграций, сложность и скорость внедрения, стоимость владения, а также гибкость в настройке и масштабировании. Это особенно важно в случае сервисных организаций, где высока зависимость от скорости отклика, точности обработки информации и адаптивности систем к внутренним регламентам.

Кроме того, сравнительный подход способствует выявлению скрытых рисков и ограничений, которые могут проявиться только в процессе эксплуатации. Например, отсутствие ИИ-механизмов или невозможность расширить сценарии в будущем может сделать систему непригодной спустя короткое время после внедрения. Анализ позволяет заранее исключить такие варианты, сохранив ресурсы и время.

Сравнительный анализ также играет ключевую роль в обосновании решения перед заинтересованными сторонами: руководством, инвесторами или техническими специалистами. Он создаёт прозрачную картину всех возможных альтернатив, показывает, почему один вариант предпочтительнее других, и помогает сформировать доверие к выбранному курсу.

Таким образом, сравнительный анализ — это не просто инструмент оценки, а стратегически важная методика, обеспечивающая осознанный, аргументированный и целесообразный выбор решений в условиях быстро меняющейся технологической среды. В рамках данной работы он стал основой для выбора архитектуры будущего виртуального помощника, подтвердив, что собственная разработка с использованием современных ИИ-инструментов представляет собой наиболее эффективный и перспективный путь для автоматизации процессов в сервисном центре [19].

Результаты сравнения указаны в таблице 1.

Таблица 1 – Результаты сравнения аналогов

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ПО | Поддержка ИИ | Интеграции | Сложность настройки | Стоимость | Гибкость настройки | Комментарий |
| *BotHelp* | Отсутствует; Только шаблоны | *Telegram*, *WhatsApp*, *Viber* | Низкая (визуальный редактор) | Условно бесплатно / от 1 300₽/мес. | Низкая | Удобен для маркетинга, слаб в автоматизации заявок |
| *Manybot* | Отсутствует | *Telegram* | Очень низкая | Бесплатно | Очень низкая | Простой, но устаревший, без ИИ и гибких сценариев |

Окончание таблицы 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ПО | Поддержка ИИ | Интеграции | Сложность настройки | Стоимость | Гибкость настройки | Комментарий |
| *Chatfuel* | Есть (через *API*) | *Telegram*, *FB*, *API* | Средняя | Условно бесплатно / от 1 900₽/мес. (для *Telegram*) | Средняя | Популярен, подходит для базовых ИИ-ботов |
| *Flow* *XO* | Есть (ограниченно) | *Telegram*, *Slack*, *API* | Средняя | От 1800₽/мес. | Средняя | Подходит для создания цепочек логики, но с ограничениями |
| *SendPulse* (*Chatbots*) | Отсутствует; (шаблоны) | *Telegram*, *VK*, *WhatsApp* | Средняя | Бесплатно до лимита (1 300₽/мес.) | Низкая | Удобен для рассылок, но слаб в логике заявок |
| *Tilda* + *Telegram* *Bot* | Отсутствует | *Telegram*, формы | Средняя | От 800₽/мес. (*Tilda*) | Низкая | Красивый фронт, но требует доработки на бэке |
| *Telegram* *Bot* + *Google* *Apps* *Script* | Есть (можно подключить) | *Telegram*, *Google* *Sheets*, *Gmail* | Высокая (код) | Бесплатно | Средняя - высокая | Хорошо масштабируется, требует навыков программирования |
| Виртуальный помощник | Есть (через *OpenAI*/*API*) | Огромное количество *API*, *Webhooks*, *CRM*, таблицы | Средняя (*no*-*code* + логика) | Бесплатно (*open*-*source*) | Очень высокая | Идеален для кастомных решений, можно получить мощную автоматизацию |

В ходе работы был проведён сравнительный анализ существующих программных решений, предназначенных для автоматизации обработки обращений в мессенджерах и на сайтах. Целью анализа было выявление наиболее подходящего инструмента, способного удовлетворить текущие и потенциальные потребности сервисного центра в части приёма, маршрутизации и обработки клиентских заявок. При сравнении учитывались ключевые параметры: наличие поддержки искусственного интеллекта, возможности интеграции с другими системами, сложность внедрения и настройки, стоимость использования, а также гибкость в адаптации под специфические бизнес-процессы.

Рассматривались такие популярные платформы, как *BotHelp*, *Manybot*, *Chatfuel*, *Flow* *XO*, *SendPulse*, связка *Tilda* и *Telegram*-ботов, а также кастомные решения на базе *Google* *Apps* *Script*. Однако большинство перечисленных инструментов либо не поддерживают интеллектуальную обработку запросов вовсе, либо предоставляют ограниченную реализацию ИИ через сторонние *API*. Их основной фокус — это маркетинговые рассылки, простые диалоги и работа с шаблонами, что делает их слабо пригодными для автоматизации сложных сценариев взаимодействия с клиентами, особенно в условиях сервисного обслуживания, где важна точность, гибкость и возможность адаптации под специфику компании.

Например, платформы *BotHelp* и *SendPulse* удобны в использовании благодаря визуальным редакторам и интеграции с мессенджерами, но имеют крайне ограниченные возможности в части логики и сценариев, практически не поддерживают сложные запросы, не способны учиться на данных и не предлагают интеллектуального анализа обращений. *Manybot*, несмотря на простоту и бесплатную модель, морально устарел и технически ограничен: отсутствие ИИ, минимальные возможности настройки и слабая поддержка интеграций делают его непригодным для современных задач.

Чуть более функциональны решения, такие как *Chatfuel* и *Flow* *XO* — они позволяют выстраивать диалоговую логику и подключать внешние *API*, однако это требует дополнительных навыков и часто сопровождается платной подпиской даже при базовом использовании. Эти платформы могут служить стартовой точкой для простых сценариев, но ограничены в масштабируемости, особенно при необходимости гибкой кастомизации под внутренние процессы сервисного центра.

Сравнительно лучше обстоит ситуация с *Telegram* *Bot* на связке с *Google* *Apps* *Script*, где возможна реализация достаточно мощной логики и интеграций. Однако и здесь высокая гибкость достигается ценой необходимости программирования, что ограничивает доступность решения для широкого круга пользователей и увеличивает нагрузку на этапах разработки и сопровождения [8].

На этом фоне наиболее перспективным решением стал собственный виртуальный помощник с использованием технологий *OpenAI* или аналогичных ИИ-платформ. Такой помощник поддерживает интеллектуальную обработку естественного языка, способен адаптироваться под конкретные задачи, легко интегрируется с внешними системами (*CRM*, *Google* *Sheets*, формы, *API* сервисов и внутренних баз), масштабируется при необходимости и может быть развёрнут без постоянных затрат — особенно если используется на основе *open*-*source* решений. Немаловажно, что его настройка может осуществляться как средствами *no*-*code*/*low*-*code* платформ, так и через прямое программирование, что делает его одновременно доступным для быстрого запуска и пригодным для глубокого кастомного развития [18].

В результате анализа можно с уверенностью утверждать, что именно реализация собственного виртуального помощника с поддержкой ИИ и возможностью интеграции с существующими цифровыми инструментами компании является наиболее эффективным выбором. Это решение обеспечивает баланс между технологической мощностью, гибкостью, экономичностью и соответствием специфике задач, стоящих перед сервисным центром ООО «Термо-Мастер сервис».

1.4 Обоснование необходимости разработки

Необходимость разработки виртуального помощника для автоматизации обработки обращений в сервисном центре ООО «Термо-Мастер сервис» обусловлена как внутренними потребностями компании, так и объективными тенденциями, характерными для современных предприятий, работающих в сфере технического обслуживания и ремонта оборудования. С каждым годом увеличивается объём информации, с которой приходится работать операторам и специалистам: растёт число клиентов, расширяется ассортимент оборудования, усложняются запросы, предъявляемые к качеству и скорости обслуживания. В таких условиях поддержание высокого уровня клиентского взаимодействия становится возможным только при использовании интеллектуальных цифровых решений, способных обрабатывать обращения быстро, точно и без привлечения избыточных человеческих ресурсов.

Сервисный центр является важнейшим звеном всей производственно-коммерческой цепочки, поскольку именно через него проходят все входящие заявки клиентов. Эти обращения могут касаться не только технических неисправностей, но и вопросов по гарантии, консультаций по эксплуатации, согласования сроков ремонта, проверки статуса заказа, оформления повторных обращений и многого другого. До настоящего момента все подобные задачи решались вручную — через телефонные звонки, мессенджеры и электронную почту. Такая модель имеет множество уязвимостей: человеческий фактор, невозможность обеспечить круглосуточную поддержку, перегрузка сотрудников, снижение скорости отклика в часы пик, ошибки при передаче информации и отсутствие унифицированного подхода к фиксации заявок.

В условиях, когда эффективность и клиентоориентированность становятся ключевыми конкурентными преимуществами, полагаться исключительно на традиционные методы взаимодействия с клиентами — значит сознательно ограничивать потенциал роста компании и создавать риск утраты лояльной аудитории. Внедрение программного виртуального помощника позволяет устранить эти ограничения за счёт автоматизации первичной коммуникации, интеллектуальной классификации запросов, мгновенного предоставления информации и оперативного сбора данных, необходимых для дальнейшего обслуживания.

Кроме того, компания планирует активно развивать цифровую инфраструктуру и стремится к минимизации времени реакции на клиентские обращения. В этом контексте виртуальный помощник становится не просто вспомогательным инструментом, а логически выверенным элементом общей стратегии цифровой трансформации сервисного направления. Благодаря использованию современных технологий обработки естественного языка и интеграции с другими системами, помощник может адаптироваться под внутренние процессы компании, обеспечивая высокую точность в интерпретации запросов, автоматическое формирование карточек обращений и распределение задач между исполнителями без участия диспетчера.

Таким образом, обоснование необходимости разработки виртуального помощника вытекает из самого характера задач, решаемых сервисным центром, и из стремления повысить уровень технологической зрелости предприятия. В условиях роста обращений, повышения требований к качеству сервиса и необходимости оптимизации внутренних ресурсов, создание такого инструмента представляется логичным, своевременным и стратегически оправданным решением, способным обеспечить устойчивое развитие компании в долгосрочной перспективе [16].

Проведённый анализ предметной области позволил выявить ключевые особенности деятельности сервисного центра ООО «Термо-Мастер сервис» и обосновать необходимость разработки виртуального помощника.

1. Анализ отрасли показал, что сервисные центры сталкиваются с высокой нагрузкой из-за большого потока обращений, необходимости оперативного реагирования и сложности ручной обработки заявок. Автоматизация процессов взаимодействия с клиентами позволит сократить время обработки запросов, минимизировать ошибки и повысить качество сервиса.
2. Анализ организации выявил, что ООО «Термо-Мастер сервис» сочетает торговую, производственную и сервисную деятельность, что создаёт сложности в управлении клиентскими обращениями. Внедрение виртуального помощника оптимизирует работу сервисного центра, разгрузит сотрудников и улучшит координацию между отделами.
3. Анализ готовых решений продемонстрировал, что большинство существующих платформ (*BotHelp*, *Manybot*, *Chatfuel* и др.) не обладают достаточной гибкостью и поддержкой ИИ для сложных сценариев. Наиболее эффективным решением является разработка собственного виртуального помощника с интеграцией современных ИИ-технологий.
4. Обоснование необходимости разработки подтверждается ростом числа обращений, потребностью в автоматизации рутинных задач и стремлением компании к цифровизации. Виртуальный помощник повысит скорость и качество обслуживания, снизит нагрузку на персонал и обеспечит конкурентное преимущество [13].

2 ПРОЕКТИРОВАНИЕ СИСТЕМЫ

2.1 Требования к системе

Разрабатываемый виртуальный помощник должен обеспечивать автоматизацию ключевых процессов взаимодействия с клиентами сервисного центра ООО «Термо-Мастер сервис», минимизируя рутинную нагрузку на сотрудников и повышая скорость обработки запросов. Основная задача системы — интеллектуальное распознавание и классификация входящих обращений, предоставление справочной информации, регистрация заявок и их передача в соответствующие подразделения компании.

Виды требований:

1. Функциональные требования.

Система должна поддерживать обработку обращений через несколько каналов коммуникации, включая мессенджеры (*Telegram*, *WhatsApp*), электронную почту и веб-интерфейс. Важным требованием является способность помощника вести осмысленный диалог с клиентом, понимать контекст запроса и уточнять детали при необходимости. Например, при поступлении заявки на ремонт система должна уметь запрашивать модель оборудования, описание неисправности и контактные данные, а также проверять наличие гарантии.

Для эффективной работы с заявками система должна интегрироваться с внутренними сервисами компании, включая *CRM*, базы данных оборудования и системы учёта заказов. Это позволит автоматически проверять статус ремонта, информировать клиентов о готовности их устройств и передавать технические спецификации в ремонтный отдел. Дополнительно помощник должен уметь формировать отчёты по частоте обращений, анализировать типовые проблемы и предоставлять статистику для оптимизации работы сервисного центра.

1. Технические требования.

Архитектура системы должна быть масштабируемой, чтобы в будущем можно было добавлять новые функции без полного перепроектирования. Поскольку обработка естественного языка требует значительных вычислительных ресурсов, важно предусмотреть возможность работы с облачными ИИ-моделями (например, *OpenAI* *GPT* или аналогичными) для обеспечения высокой точности распознавания запросов. Система также должна поддерживать *API*-интеграции с внешними сервисами, такими как *Google* *Sheets* для ведения учёта или платежные системы для онлайн-оплаты услуг.

1. Безопасность и надёжность.

Учитывая, что помощник будет обрабатывать персональные данные клиентов и информацию о заказах, необходимо обеспечить защиту передаваемых данных с помощью шифрования и строгой аутентификации пользователей. Система должна быть отказоустойчивой: в случае сбоев или высокой нагрузки важно предусмотреть механизмы сохранения незавершённых диалогов и их восстановления.

1. Пользовательский опыт.

Интерфейс взаимодействия с виртуальным помощником должен быть интуитивно понятным как для клиентов, так и для сотрудников сервисного центра. Для клиентов это означает простоту формулирования запросов, минимальное количество шагов для регистрации заявки и оперативность ответов. Для администраторов системы важна удобная панель управления, позволяющая настраивать сценарии диалогов, корректировать базу знаний и отслеживать выполнение заказов.

1. Перспективы развития.

В будущем систему можно расширить за счёт внедрения голосового интерфейса, поддержки мультиязычности (например, для обслуживания иностранных клиентов) или подключения машинного обучения для автоматического анализа причин неисправностей оборудования. Однако на первом этапе важно обеспечить стабильную работу базовых функций, чтобы виртуальный помощник стал неотъемлемой частью сервисного центра, а не дополнительным инструментом, требующим постоянного контроля со стороны персонала [17].

Таким образом, виртуальный помощник должен сочетать в себе гибкость, надёжность и удобство, становясь не просто автоматизированным ответчиком, а полноценным элементом клиентского сервиса компании.

2.2 Функциональная модель системы

Функциональная модель системы представляет собой формализованное описание её работы, которое определяет, какие задачи система должна выполнять, как она взаимодействует с пользователями и другими компонентами, а также какие процессы и данные задействованы в её работе. В отличие от технической архитектуры, которая описывает «как система устроена», функциональная модель отвечает на вопрос «что система делает» [20].

В случае виртуального помощника для сервисного центра функциональная модель описывает:

1. Основные сценарии взаимодействия (например, приём заявки, консультация клиента, проверка статуса ремонта);
2. Потоки данных (как информация передаётся между клиентом, помощником и внутренними системами компании);
3. Бизнес-логику (правила обработки запросов, автоматические действия и исключительные ситуации);
4. Интеграции с внешними сервисами (*CRM*, базы данных, мессенджеры).

Функциональная модель помогает:

1. Чётко определить границы системы – что входит в её зону ответственности, а что остаётся за её пределами (например, сам ремонт оборудования система не выполняет, но фиксирует заявку на него).
2. Увидеть взаимосвязи между процессами – как, например, запрос клиента в *Telegram* превращается в заявку в *CRM* и уведомление для мастера.
3. Выявить «узкие места» – ситуации, где возможны ошибки или задержки (например, если клиент не указал модель устройства, система должна уметь запросить недостающие данные).
4. Согласовать требования между заказчиком и разработчиками – модель служит «мостиком» между бизнес-задачами и технической реализацией.

Для визуализации функциональной модели часто используют:

1. Диаграммы *Use* *Case* (кто и какие действия выполняет);
2. *BPMN*-схемы (последовательность процессов);
3. Пользовательские сценарии (пошаговые примеры работы системы).

Пример функциональной модели виртуального помощника:

1. Клиент отправляет сообщение «Мой ноутбук не включается».
2. Система распознаёт намерение («ремонт»), запрашивает модель и гарантийный статус.
3. После заполнения данных создаёт заявку в *CRM* и уведомляет мастера.
4. При изменении статуса ремонта автоматически информирует клиента.

Таким образом, функциональная модель – это «инструкция» для системы, которая объясняет не её устройство, а её поведение в реальных условиях. Без такой модели разработка рискует превратиться в набор разрозненных функций, а не в целостный инструмент для бизнеса [5].

Диаграмма прецедентов проектируемой системы представлена на рисунке 2.

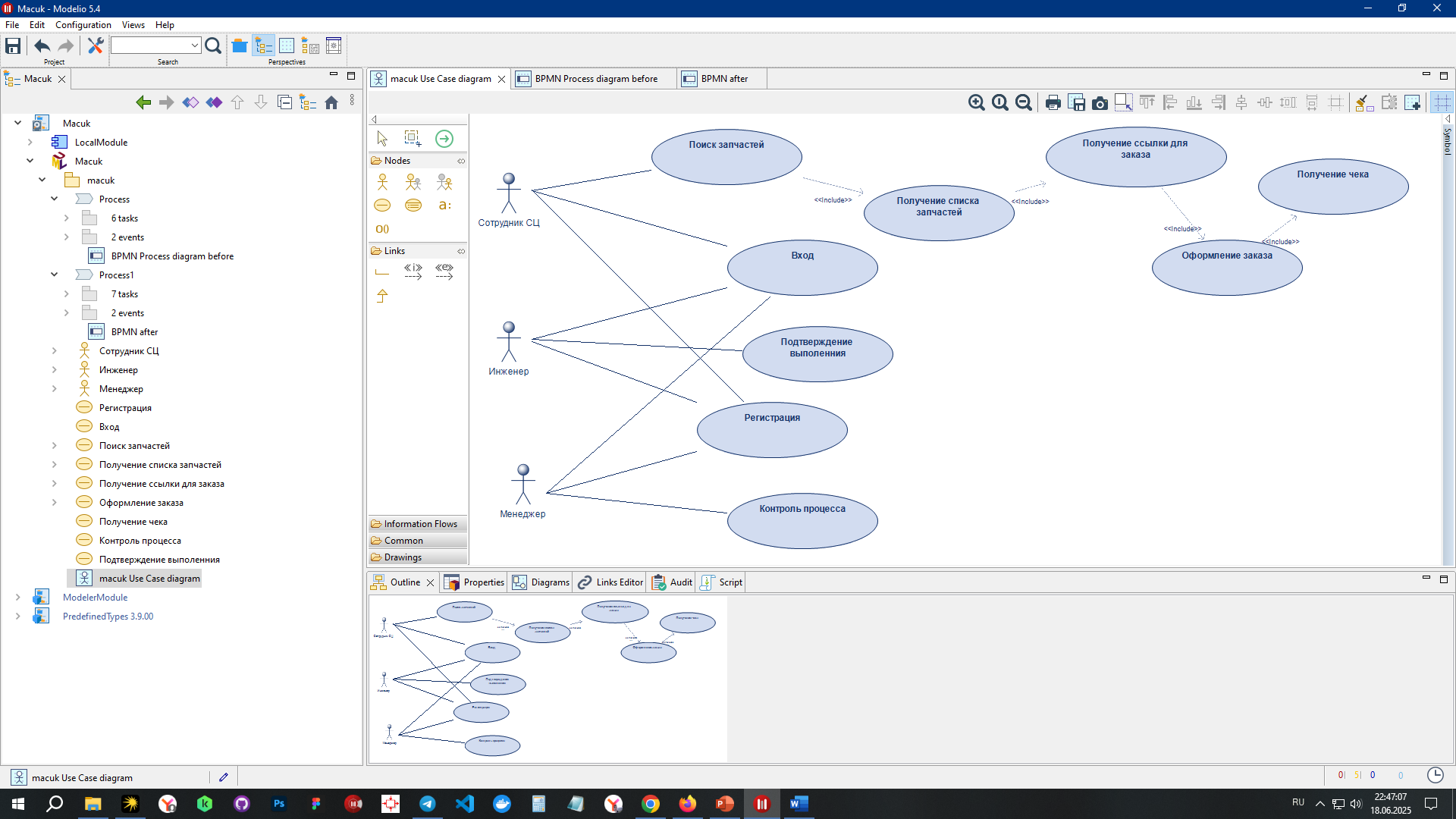


Рисунок 2 – Диаграмма прецедентов

Диаграмма вариантов использования представляет собой визуальное отображение взаимодействий между актором и системой, в данном случае — «Сотрудником Сервисного Центра» (СЦ). Эта диаграмма позволяет наглядно представить функциональные требования к системе, а также способы, которыми пользователи могут взаимодействовать с ней.

Актор «Сотрудник СЦ» является центральной фигурой в данной диаграмме. Он осуществляет несколько ключевых действий, которые обеспечивают эффективное выполнение задач, связанных с обработкой заказов и управлением запасами.

Варианты использования:

1. Регистрация - сотрудник СЦ имеет возможность зарегистрироваться в системе, что позволяет ему получать доступ к функционалу и выполнять необходимые операции.
2. Вход - после регистрации сотрудник может входить в систему, используя свои учетные данные, что обеспечивает безопасность и контроль доступа.
3. Поиск запчастей этот вариант использования позволяет сотруднику осуществлять поиск необходимых запчастей в базе данных, что ускоряет процесс обслуживания клиентов и повышает эффективность работы.
4. Получение списка запчастей (<<*include*>>): Сотрудник может получать список доступных запчастей, что является важным шагом для дальнейшего оформления заказов. Этот процесс включен в общий сценарий использования, так как он напрямую связан с другими действиями.
5. Получение ссылки для заказа - после нахождения необходимых запчастей сотрудник получает ссылку для оформления заказа, что делает процесс более удобным и быстрым.
6. Оформление заказа (<<*include*>>) - данный вариант использования подразумевает, что сотрудник может оформлять заказы на запчасти, которые он нашел ранее. Это также включает в себя взаимодействие с другими компонентами системы для завершения заказа.
7. Получение чека (<<*include*>>) - после успешного оформления заказа, сотрудник получает чек, который служит подтверждением завершенной транзакции.

Данная диаграмма вариантов использования наглядно демонстрирует основные функции, которые доступны сотруднику СЦ, а также последовательность действий, необходимых для выполнения его задач. Это позволяет не только систематизировать процесс, но и выявить потенциальные области для улучшения в дальнейшем.

*BPMN*-схемы представляют собой графический стандарт для моделирования бизнес-процессов, позволяющий визуализировать последовательность действий, участников и логику работы системы в понятной и стандартизированной форме. Этот подход помогает проектировать, анализировать и оптимизировать процессы, делая их прозрачными для всех участников - от бизнес-аналитиков до разработчиков и конечных пользователей.

Основу *BPMN*-схем составляют ключевые элементы, каждый из которых выполняет конкретную функцию в описании процесса. Действия, обозначаемые прямоугольниками со скругленными углами, показывают конкретные шаги, которые необходимо выполнить. События, изображаемые в виде кругов с различными иконками, отмечают важные моменты - начало, промежуточные этапы или завершение процесса. Шлюзы в форме ромбов помогают отображать точки принятия решений и ветвления логики. Потоки, представленные стрелками, связывают все элементы в единую последовательность, а дорожки разделяют ответственность между разными участниками процесса.

Главная ценность *BPMN* заключается в его универсальности и наглядности. Используя этот стандарт, можно превратить сложные многоэтапные процессы в понятные схемы, которые легко анализировать и совершенствовать. Например, при разработке виртуального помощника *BPMN* позволяет детально описать, как система должна реагировать на различные типы запросов пользователей, куда и в каком формате передавать информацию, как обрабатывать исключительные ситуации. Это особенно важно при интеграции помощника с другими корпоративными системами, такими как *CRM* или базы данных.

Отличительной особенностью *BPMN* по сравнению с другими методами моделирования является его ориентация именно на бизнес-процессы, а не на техническую реализацию. В отличие от *UML*, который фокусируется на структуре программного обеспечения, *BPMN* описывает логику работы с точки зрения бизнес-задач. По сравнению с обычными блок-схемами, *BPMN* предлагает гораздо более богатый набор элементов для отображения сложных сценариев, включая параллельные процессы, временные события и различные типы взаимодействий [20].

На практике применение *BPMN* для виртуального помощника может выглядеть следующим образом: схема начинается с события получения сообщения от пользователя, затем следует серия действий по анализу запроса и уточнению информации, после чего процесс может разветвляться в зависимости от типа обращения - создание заявки на ремонт, предоставление справочной информации или проверка статуса существующего заказа. Каждый вариант предполагает свою последовательность действий с участием разных систем и специалистов, что наглядно отображается в схеме.

Важно отметить, что *BPMN* не только помогает в проектировании, но и служит эффективным инструментом коммуникации между различными подразделениями компании. Технические специалисты на основе этих схем могут разрабатывать архитектуру решения, бизнес-аналитики - выявлять возможности для оптимизации, а руководители - оценивать эффективность процессов. Кроме того, многие современные системы автоматизации бизнес-процессов позволяют напрямую импортировать *BPMN*-схемы, ускоряя таким образом внедрение новых решений.

Таким образом, *BPMN*-схемы представляют собой мощный инструмент для работы с бизнес-процессами любого уровня сложности. Их использование при создании виртуального помощника позволяет добиться четкого понимания того, как система должна функционировать, как она будет взаимодействовать с пользователями и другими компонентами инфраструктуры, а также какие варианты развития процессов необходимо предусмотреть на этапе проектирования [12].

Процесс заказа запчастей без внедрения виртуального помощника представлен на рисунке 3.

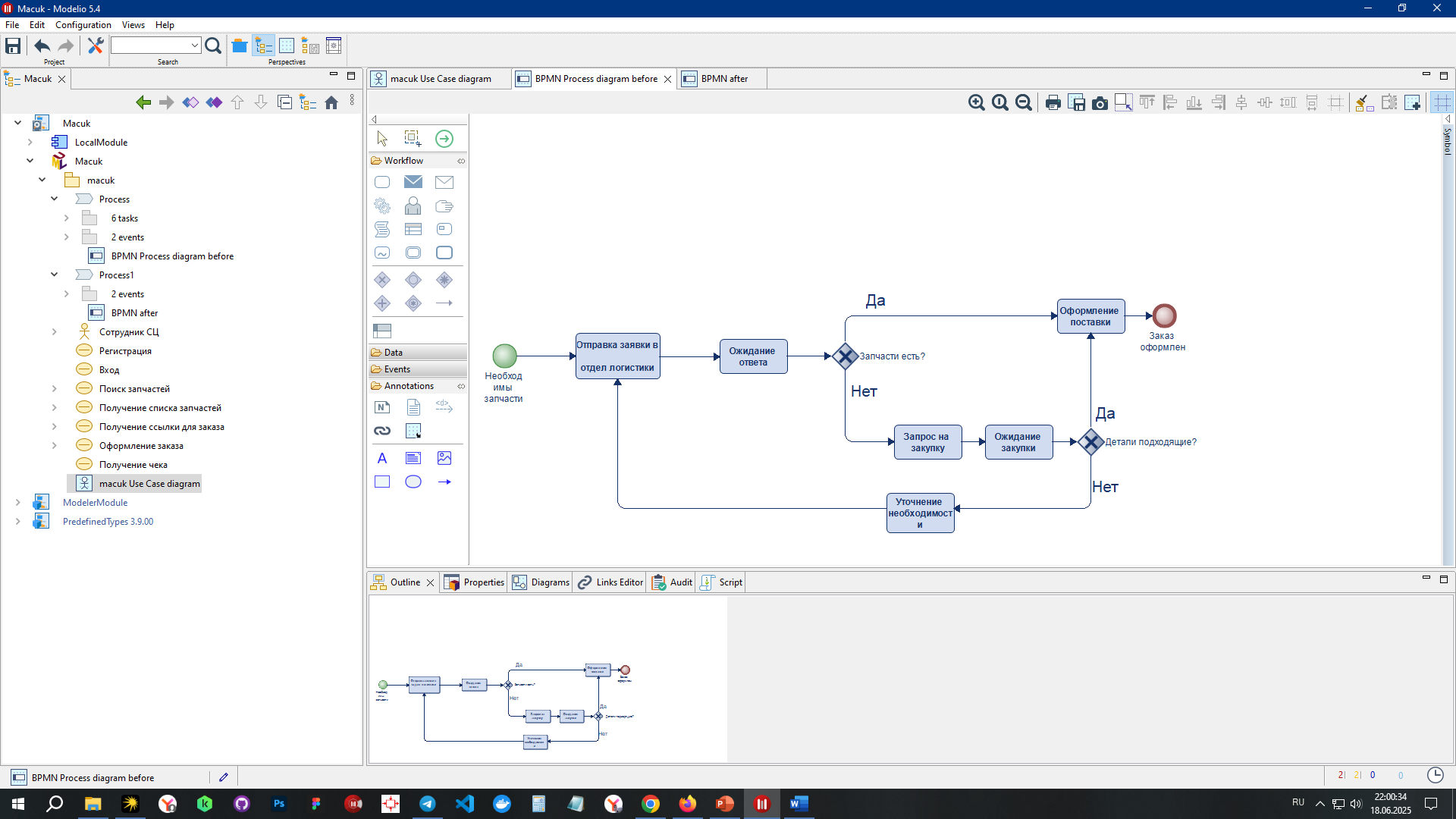


Рисунок 3 – Процесс заказа запчастей до внедрения виртуального помощника

Процесс заказа запчастей, как описано в диаграмме *BPMN*, представляет собой последовательность действий, которые начинают с момента, когда возникают потребности в запчастях. Этот процесс начинается с этапа «Необходимы запчасти», где сотрудник определяет, какие именно детали нужны для дальнейшей работы.

Основные этапы процесса:

1. Отправка заявки в отдел логистики - после определения необходимых запчастей, заявка направляется в логистический отдел для обработки. Это первый шаг, который запускает дальнейшие действия.
2. Ожидание ответа: В этот момент процесс может столкнуться с задержками, так как необходимо дождаться ответа от логистики. Это время ожидания может негативно сказаться на общей производительности.
3. Решение: «Запчасти есть?» - на этом этапе принимается решение, которое определяет дальнейшие действия. Если запчасти имеются, процесс продолжается с «Оформления поставки» и завершается «Заказом оформленным». Однако, если запчастей нет, начинается более сложный путь.
4. Запрос на закупку - в случае отсутствия необходимых деталей, создается запрос на закупку. Это добавляет дополнительный уровень сложности и времени к процессу.
5. Ожидание закупки - как и в предыдущем этапе, процесс сталкивается с временными задержками, так как необходимо дождаться поступления новых запчастей.
6. Решение: «Детали подходящие?» - после поступления новых запчастей необходимо снова принять решение. Если детали удовлетворяют требованиям, процесс продолжается с «Оформлением поставки». Если же они не подходят, начинается «Уточнение необходимости».

Данный процесс, несмотря на его структурированность, требует оптимизации для повышения его эффективности. Временные задержки на этапах ожидания ответов и закупки могут значительно замедлить выполнение задач и, как следствие, повлиять на общую производительность компании.

Рекомендуется рассмотреть возможность автоматизации некоторых этапов, таких как:

1. Автоматизированная система уведомлений: для сокращения времени ожидания ответов от логистики.
2. Интеграция с поставщиками: чтобы минимизировать задержки при закупке запчастей и ускорить процесс получения нужных деталей.
3. Оптимизация принятия решений: внедрение более четких критериев для выбора подходящих запчастей, чтобы уменьшить количество возвратов к этапу «Уточнение необходимости».

Оптимизация этих процессов не только повысит скорость реакции на запросы, но и улучшит общее качество обслуживания клиентов, что является важным аспектом в конкурентной среде [15].

Процесс заказа запчастей после внедрения виртуального помощника представлен на рисунке 4.

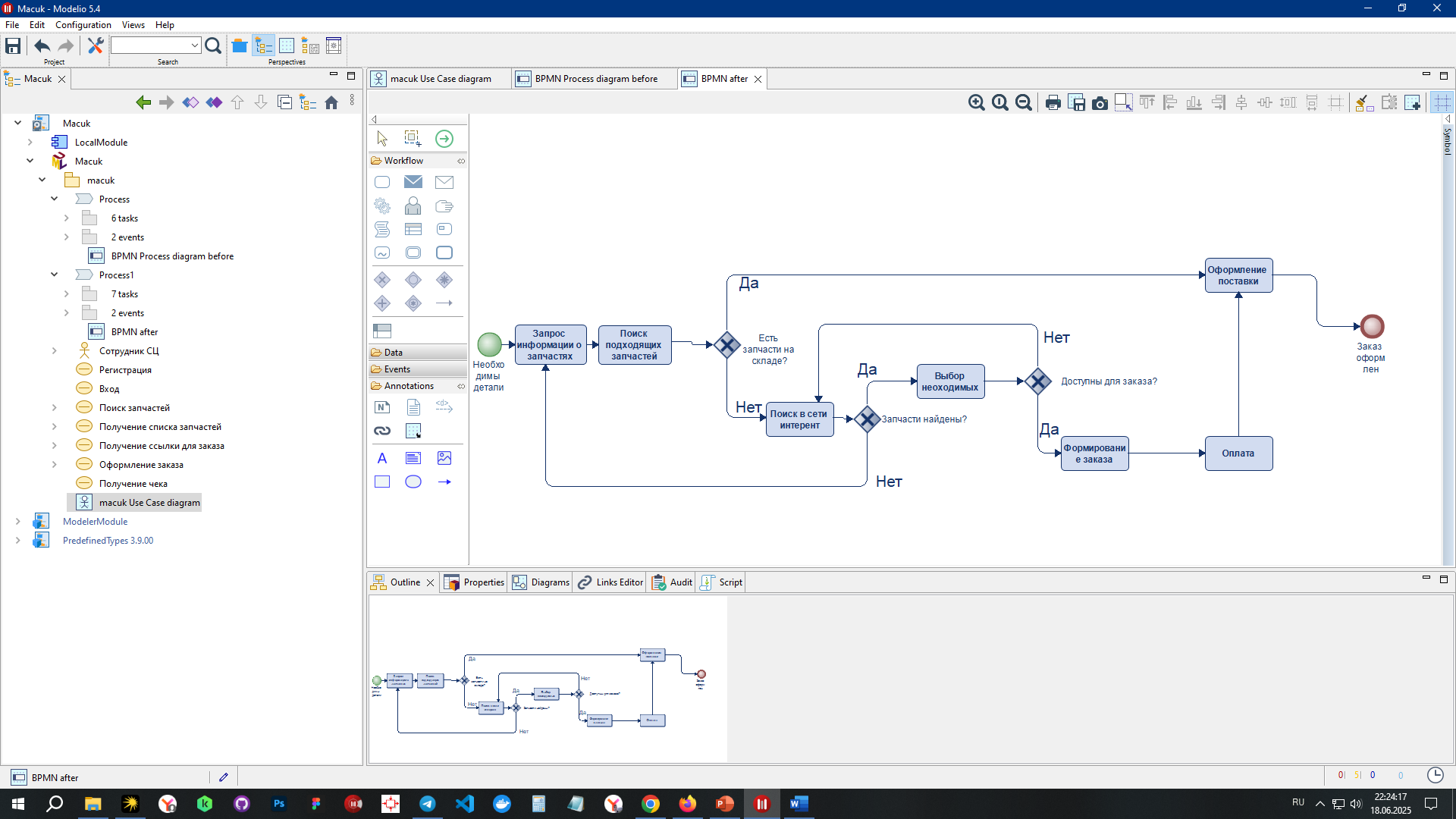


Рисунок 4 – Процесс заказа запчастей после внедрения виртуального помощника

С внедрением виртуального помощника процесс заказа запчастей претерпел значительные изменения, что сделало его более эффективным и быстрым. Теперь процесс начинается с этапа определения необходимых деталей, что позволяет сотруднику сразу приступить к запросу информации о запчастях.

Следующим шагом является автоматизированный поиск подходящих запчастей на складе. Виртуальный помощник мгновенно обрабатывает запрос и определяет наличие деталей. Если запчасти имеются, процесс переходит к формированию заказа. В случае их отсутствия помощник автоматически инициирует поиск в интернете, что значительно сокращает временные затраты на ручной поиск.

На этапе поиска в сети интернет виртуальный помощник также проверяет доступность запчастей для заказа. Это позволяет избежать ненужных задержек и быстрее переходить к выбору необходимых деталей. После выбора виртуальный помощник формирует заказ, включая автоматическую проверку на предмет наличия всех необходимых компонентов.

Следующий шаг — это оплата, который также становится более упрощенным за счет интеграции с различными платежными системами. Виртуальный помощник обеспечивает безопасный и быстрый процесс оплаты, что значительно улучшает пользовательский опыт.

Завершающим этапом является оформление поставки и подтверждение заказа. Благодаря автоматизации этого процесса, клиенты получают уведомления о статусе своего заказа в реальном времени, что повышает уровень удовлетворенности и доверия.

В результате внедрения виртуального помощника процесс заказа запчастей стал не только оптимизированным, но и более прозрачным, что позволяет сократить время выполнения заказов и повысить общую эффективность работы [14].

Декомпозиция общей диаграммы, процесс «Запрос информации о запчастях» представлен на рисунке 5.

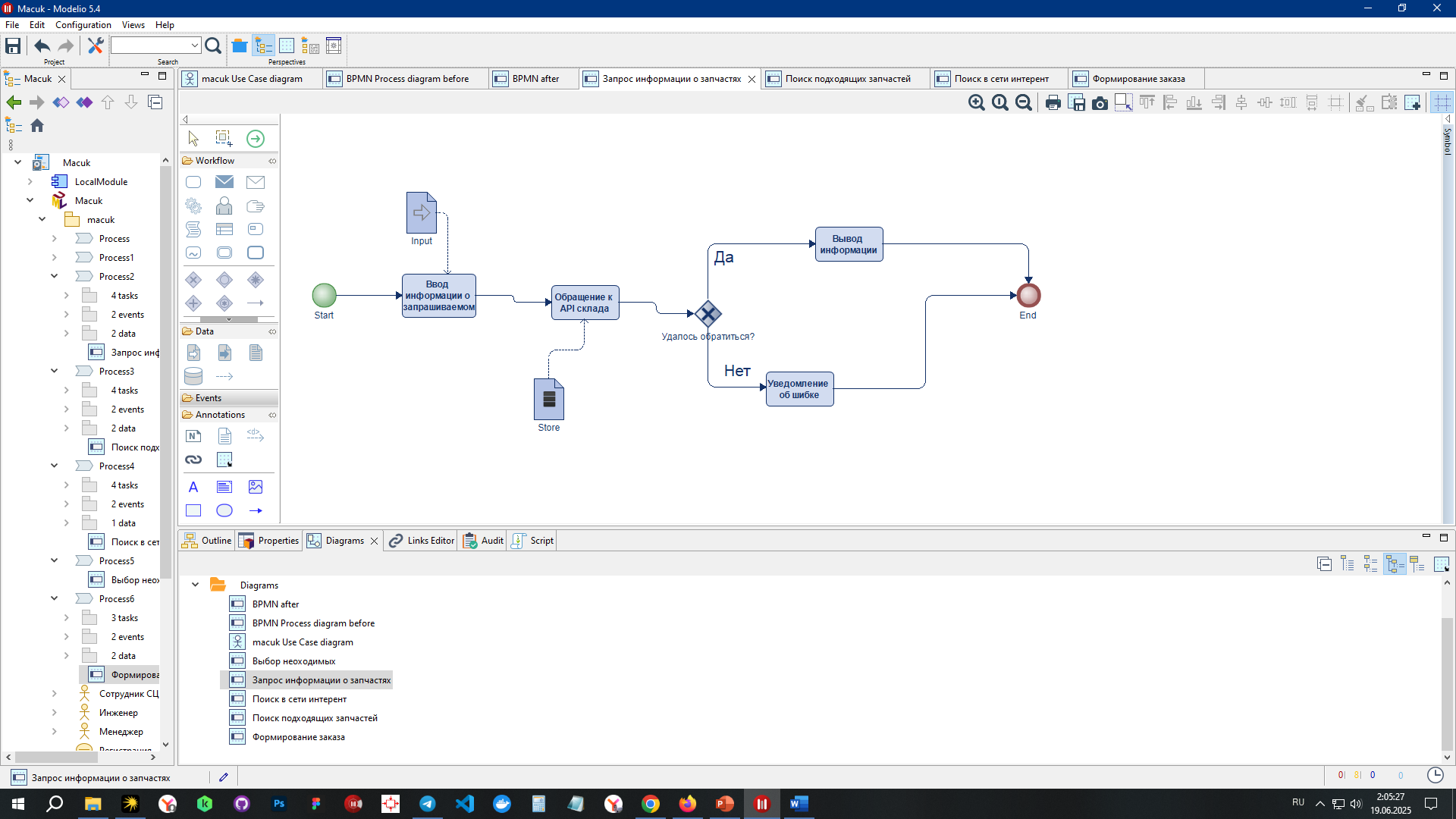


Рисунок 5 – Процесс «Запрос информации о запчастях»

Диаграмма *BPMN* «Запрос информации о запчастях» начинается с получения запроса пользователя через *Telegram*-бота. Виртуальный помощник анализирует запрос и проверяет наличие запчасти на складе. Если деталь найдена, бот отправляет информацию о наличии и предлагает оформить заказ. Если запчасти нет, бот просит пользователя уточнить запрос или предлагает альтернативы. При повторном уточнении запрос проверяется заново. Если деталь так и не найдена, процесс переходит к поиску в сторонних интернет-магазинах. Такой цикл обеспечивает гибкое и эффективное взаимодействие пользователя с системой.

Декомпозиция общей диаграммы, процесс «Поиск подходящих запчастей» представлен на рисунке 6.

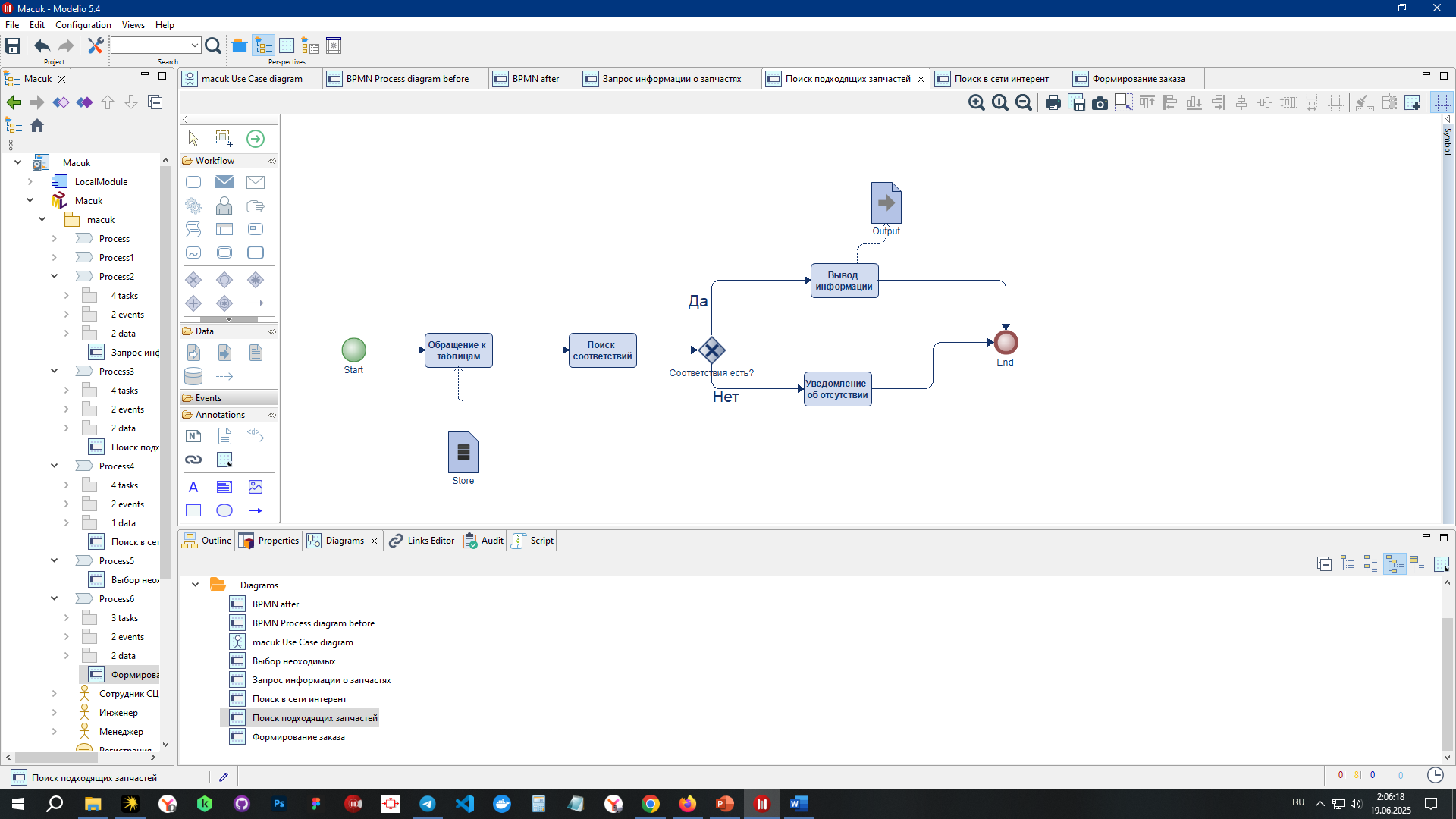


Рисунок 6 – Процесс «Поиск подходящих запчастей»

Диаграмма *BPMN* «Поиск подходящих запчастей» начинается с получения запроса пользователя на определённую деталь. Виртуальный помощник анализирует параметры и сначала проверяет наличие запчасти на складе. Если нужной позиции нет, бот автоматически инициирует поиск аналогичных или совместимых запчастей среди внешних поставщиков и интернет-магазинов. После сбора доступных вариантов бот формирует список с ценами и ссылками, который отправляет пользователю для выбора. Пользователь выбирает подходящий вариант, после чего бот предлагает оформить заказ. Этот процесс обеспечивает быстрый и удобный подбор запчастей с максимальным охватом источников.

Декомпозиция общей диаграммы, процесс «Поиск в сети интернет» представлен на рисунке 7.



Рисунок 7 – Процесс «Поиск в сети интернет»

Диаграмма *BPMN* «Поиск в сети интернет» начинается с того, что виртуальный помощник получает запрос пользователя на запчасть, которая отсутствует на складе. После подтверждения отсутствия детали, бот запускает автоматизированный поиск по доступным интернет-магазинам и онлайн-платформам, собирая информацию о наличии, ценах и условиях доставки. Затем формируется список найденных предложений, который отправляется пользователю для ознакомления и выбора. После выбора подходящего варианта бот предлагает оформить заказ или перейти к оплате. Этот процесс позволяет оперативно найти и предложить альтернативные источники запчастей вне внутренних ресурсов компании.

Декомпозиция общей диаграммы, процесс «Формирование заказа» представлен на рисунке 8.

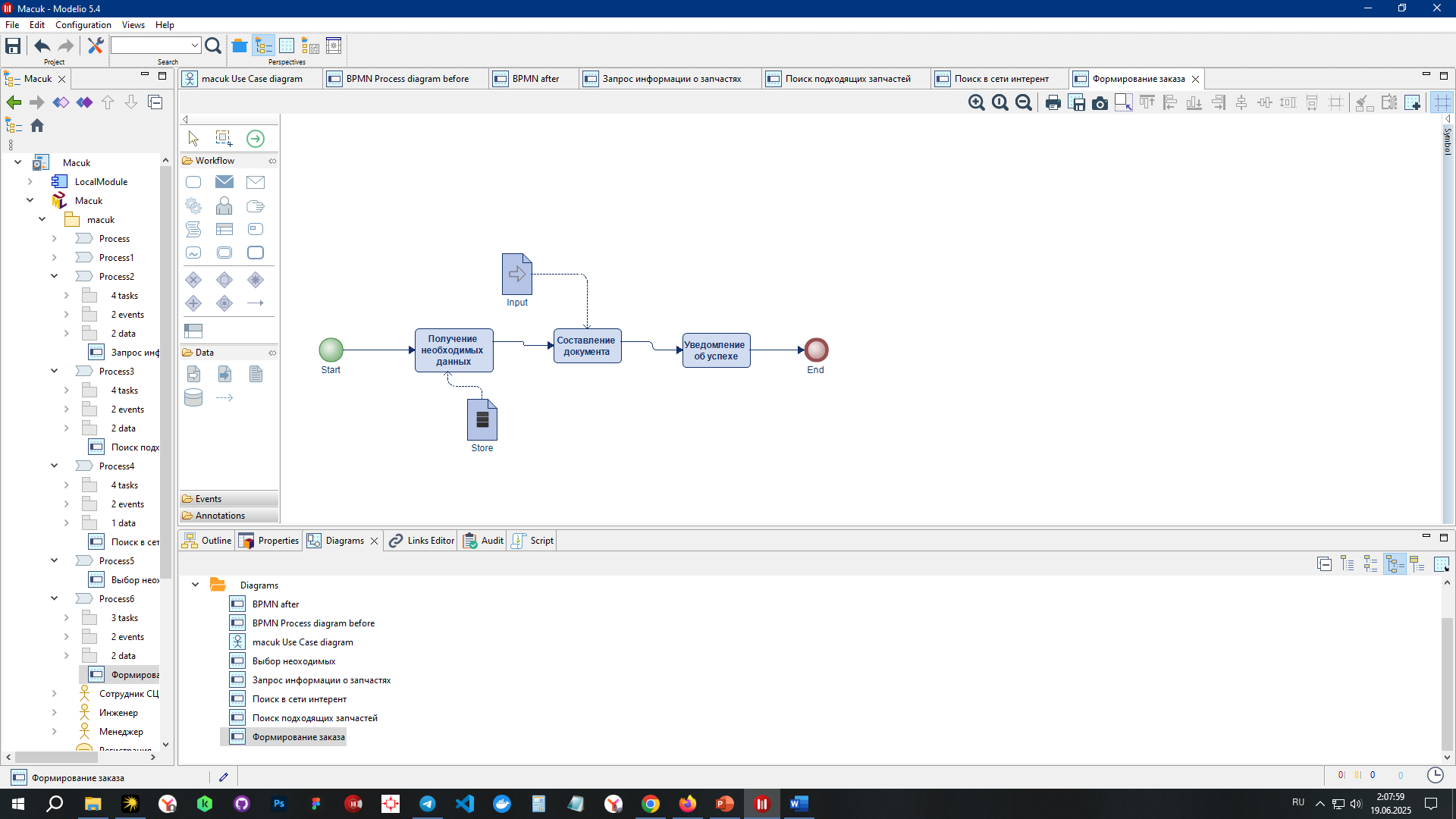


Рисунок 8 – Процесс «Формирование заказа»

Диаграмма *BPMN* «Поиск в сети интернет» описывает процесс, который начинается после того, как виртуальный помощник получает запрос пользователя на запчасть, отсутствующую на складе компании. Система фиксирует факт отсутствия товара и автоматически инициирует поиск на внешних ресурсах, включая интернет-магазины, торговые площадки и поставщиков. Для этого бот обращается к интегрированным *API*, веб-сервисам и базам данных партнеров, собирая актуальную информацию о наличии товара, ценах, условиях доставки и сроках поставки.

После получения и обработки данных формируется структурированный список доступных вариантов, который содержит наименование запчасти, цену, ссылки на сайты продавцов и другую важную информацию. Этот список отправляется пользователю через интерфейс бота для ознакомления и выбора наиболее подходящего предложения. В случае необходимости бот может дополнительно предоставить рекомендации или фильтры для удобства выбора.

Когда пользователь определяет предпочтительный вариант, виртуальный помощник предлагает оформить заказ, предоставляя информацию о способах оплаты и дальнейших шагах. Система фиксирует выбор, генерирует необходимые документы и, при интеграции с внутренними учетными системами, передает информацию в отдел закупок или логистики для дальнейшей обработки. По окончании процесса пользователю направляется уведомление с подтверждением заказа, деталями оплаты и информацией о доставке [12].

Данный процесс обеспечивает оперативное и эффективное расширение возможностей поиска нужных запчастей за пределами внутреннего склада, позволяя сократить время ожидания и повысить уровень сервиса для клиентов и сотрудников сервисного центра.

2.3 Архитектура системы

Архитектура системы это совокупность структурных элементов программного обеспечения и принципов их взаимодействия, которая определяет организацию всех компонентов системы, способы передачи данных между ними, а также правила, по которым осуществляется обработка информации и выполнение функциональных задач. Она служит фундаментом, на котором строится вся система, и задаёт рамки для последующей разработки, интеграции, масштабирования и сопровождения программного продукта.

На практике архитектура определяет, из каких логических и технических блоков состоит система, какие модули будут реализованы, каким образом они будут взаимодействовать друг с другом, какие интерфейсы и протоколы будут использоваться, где и как будут храниться данные, как будет обеспечиваться безопасность, отказоустойчивость, доступность и производительность. Иными словами, архитектура это не просто схема, а стратегическое описание «скелета» всей системы, обеспечивающее её целостность и управляемость на всех этапах жизненного цикла.

Важной задачей архитектуры является также разделение ответственности между компонентами. Это позволяет сделать систему гибкой, облегчить отладку и модернизацию, а также адаптировать её под изменения требований. Например, можно выделить интерфейсный слой, где происходит взаимодействие с пользователем, логический слой, отвечающий за принятие решений и обработку команд, и слой хранения данных, обеспечивающий доступ к базе и сохранение информации. Такая модульность позволяет изменять или дорабатывать отдельные части системы без необходимости полного её переписывания.

Кроме того, архитектура определяет способ развертывания системы — будет ли она монолитной, когда всё приложение собирается в единое целое, или распределённой, где каждая функциональная часть работает независимо, возможно, на разных серверах или даже в разных географических точках. В современных условиях всё чаще используются распределённые и микросервисные архитектуры, которые обеспечивают масштабируемость, устойчивость к сбоям и возможность гибкой настройки под конкретные бизнес-потребности.

В случае разработки виртуального помощника для сервисного центра выбор и описание архитектуры особенно важны, поскольку необходимо учитывать интеграцию с внешними системами (например, *CRM* или базой заказов), поддержку обработки естественного языка, обмен сообщениями в реальном времени, хранение логов и историю общения, а также возможность масштабирования при увеличении числа обращений. Грамотно спроектированная архитектура в этом случае позволяет создать устойчивое, надёжное и эффективное программное решение, которое сможет развиваться вместе с компанией и адаптироваться к новым требованиям без кардинальной переработки всей системы.

Таким образом, архитектура системы — это не просто технический термин, а ключевая концепция, определяющая жизнеспособность, удобство и эффективность всего программного решения. От правильности архитектурных решений зависит, насколько легко будет развивать, сопровождать и использовать систему в реальных условиях [18].

Архитектура система представлена на рисунке 9.

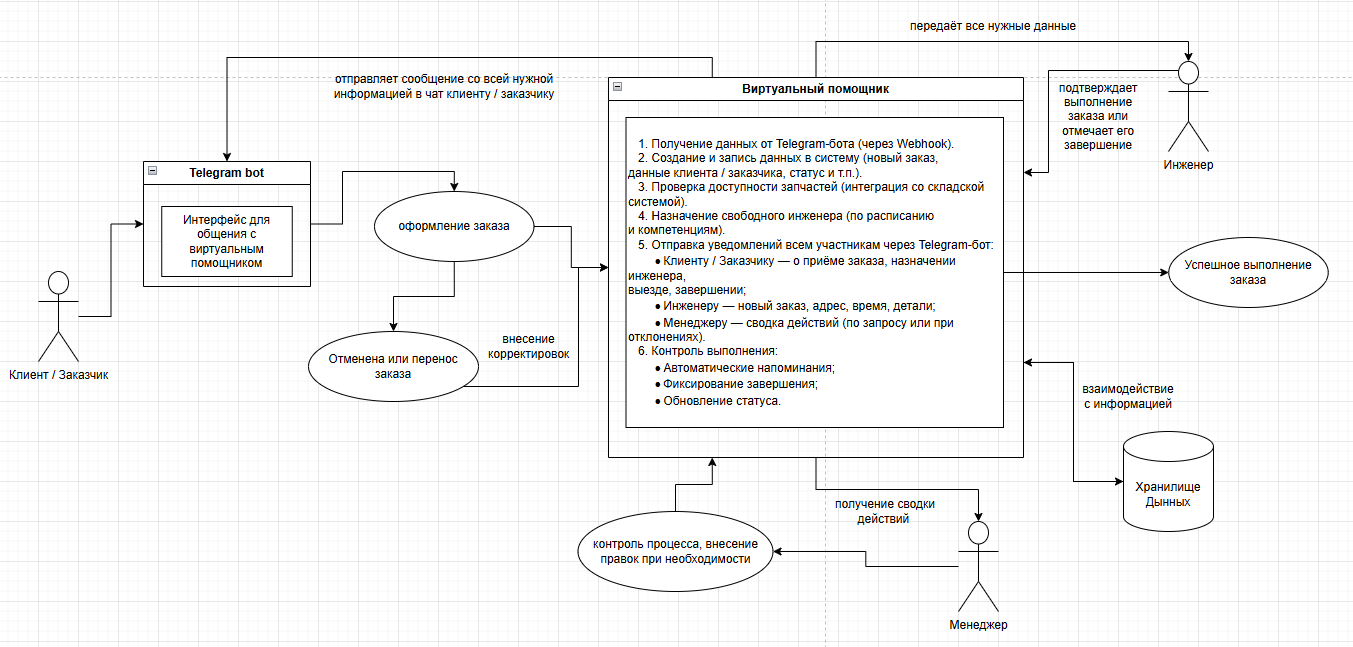


Рисунок 9 – Архитектура проектируемой системы

Проектируемая система автоматизированного заказа представляет собой интегрированное решение, которое включает в себя несколько ключевых компонентов: Клиента/Заказчика, *Telegram*-бота, Виртуального помощника, Инженера, Менеджера и Хранилище данных. Эта архитектура направлена на оптимизацию процесса заказа и обеспечения высокой эффективности взаимодействия между всеми участниками.

Основные компоненты системы:

1. Заказчик является инициатором процесса заказа. Он взаимодействует с системой через *Telegram*-бота, отправляя необходимую информацию и получая уведомления о статусе своего заказа.
2. *Telegram*-бот выполняет роль интерфейса для общения между Клиентом и Виртуальным помощником. Бот позволяет клиентам удобно оформлять заказы, получать статус и вносить корректировки. Он также обеспечивает отправку уведомлений о состоянии заказа.
3. Виртуальный помощник центральный элемент архитектуры, который обрабатывает входящие запросы. Он получает данные от *Telegram*-бота через *Webhook* и создает записи в системе, включая новые заказы и данные клиента. Виртуальный помощник проверяет доступность запчастей, назначает инженера, отправляет уведомления всем участникам и контролирует выполнение заказа с помощью автоматических напоминаний и обновления статуса.
4. Инженер важный игрок в процессе, который подтверждает выполнение заказа и отмечает его завершение. Инженер получает уведомления о новых заказах, включая адрес, время и необходимые детали.
5. Менеджер отвечает за контроль процесса и взаимодействие с информацией. Он получает сводки действий по запросу или при отклонениях, что позволяет ему эффективно управлять рабочими процессами.
6. Хранилище данных обеспечивает хранение всей необходимой информации о заказах, клиентах и статусах. Оно передает данные всем участникам системы, обеспечивая их актуальность и доступность.

Архитектура системы направлена на максимальную автоматизацию и оптимизацию всех этапов заказа. Внедрение виртуального помощника позволяет существенно сократить время обработки заказов, минимизировать риски ошибок и повысить уровень удовлетворенности клиентов. Система обеспечивает непрерывный контроль выполнения заказов и позволяет вносить корректировки в режиме реального времени, что значительно улучшает качество обслуживания.

Таким образом, данная архитектура систематизирует и упрощает процесс заказа запчастей, позволяя всем участникам взаимодействовать более эффективно [6].

2.4 Моделирование данных

Моделирование данных — это процесс создания абстрактного представления информации, которое отражает структуру, взаимосвязи и правила организации данных в рамках определённой предметной области или системы. Этот процесс позволяет понять, как именно данные должны быть организованы, чтобы эффективно храниться, обрабатываться и использоваться для решения бизнес-задач. Моделирование данных играет ключевую роль в разработке программного обеспечения, баз данных и информационных систем, поскольку именно от качества модели зависит удобство работы с данными, их целостность и возможность масштабирования.

В ходе моделирования создаются различные схемы и диаграммы, которые описывают объекты и сущности, их свойства, а также связи между ними. Это помогает выявить основные компоненты системы и прояснить, каким образом они взаимодействуют друг с другом. Модели данных выступают в качестве «карты» для разработчиков и аналитиков, позволяя согласовать требования и обеспечить единое понимание структуры данных на всех этапах разработки.

Моделирование данных позволяет упорядочить огромные объёмы информации, выделить ключевые элементы и задать правила для хранения и обработки, что особенно важно при создании баз данных и построении сложных систем. Оно способствует выявлению избыточности, аномалий и несоответствий в данных, что помогает оптимизировать структуру и повысить качество данных. Кроме того, моделирование облегчает интеграцию различных систем, поскольку обеспечивает стандартизированное описание данных, понятное для всех участников процесса.

Существует несколько видов моделирования данных, среди которых наиболее распространены концептуальное, логическое и физическое моделирование. Концептуальное моделирование фокусируется на высокоуровневом описании предметной области без привязки к техническим деталям. Логическое моделирование углубляется в структуру данных, учитывая требования к их организации, но остаётся независимым от конкретных систем управления базами данных. Физическое моделирование отражает конкретную реализацию модели в используемой системе, включая специфику хранения и доступа к данным.

Таким образом, моделирование данных — это фундаментальный этап проектирования информационных систем, который обеспечивает создание точной, понятной и оптимальной структуры данных. Оно способствует эффективной реализации программных решений, облегчает их сопровождение и развитие, а также повышает надёжность и производительность систем за счёт грамотной организации информации [17].

Модель данных проектируемой системы представлена на рисунке 10.

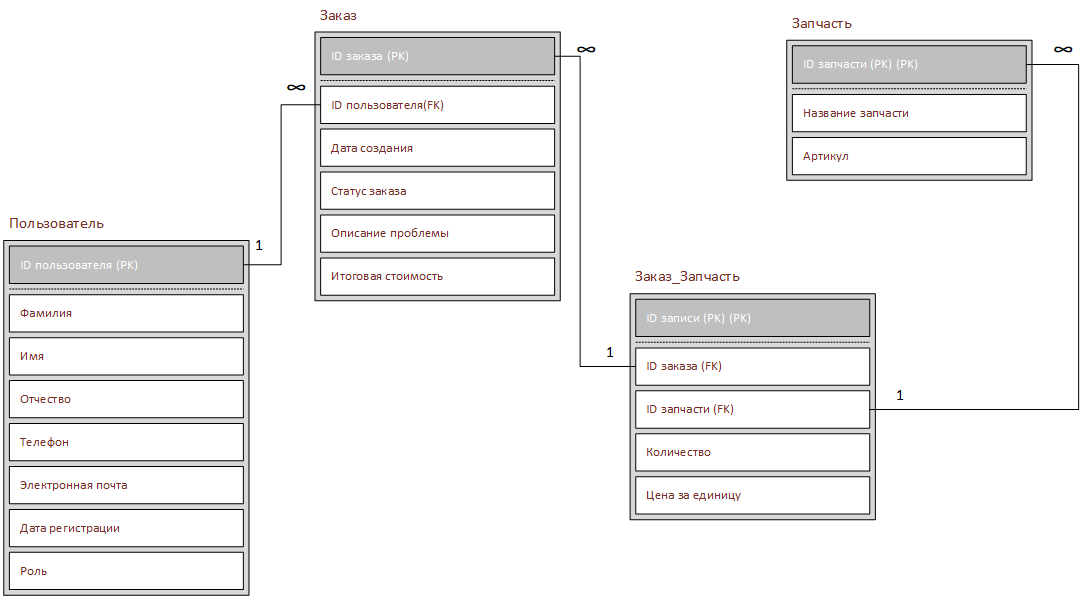


Рисунок 10 – Модель данных проектируемой системы

В структуре данных выделены три основные сущности — пользователь, заказ и запчасть, между которыми установлены важные взаимосвязи, отражающие реальную логику работы сервисного центра. Один пользователь может иметь несколько заказов, что характеризует отношение один-ко-многим, позволяя учитывать все обращения и заявки конкретного клиента в системе. Каждый заказ, в свою очередь, может включать множество запчастей, необходимых для выполнения работ или ремонта, что также является связью один-ко-многим. Для реализации этой связи используется дополнительная промежуточная таблица, которая связывает конкретные заказы с конкретными запчастями, позволяя фиксировать количество и цену каждой позиции в рамках заказа.

Таблица пользователей содержит уникальные идентификаторы и основные контактные данные, что обеспечивает возможность точной идентификации клиентов и назначения им заказов. В таблице заказов хранится информация о каждом обращении — дата создания, текущий статус, описание проблемы и итоговая стоимость. Такой подход позволяет не только регистрировать обращения, но и отслеживать их состояние на различных этапах обработки, а также анализировать активность пользователей.

Таблица запчастей содержит данные о доступных деталях, включая их наименование, артикулацию, количество на складе и цену. Связь между заказами и запчастями реализована через дополнительную таблицу, которая учитывает, какие именно детали и в каком количестве были использованы для конкретного заказа. Это обеспечивает точный учёт ресурсов и позволяет оптимизировать процесс ремонта и обслуживания [17].

Таким образом, представленная модель данных отражает ключевые бизнес-процессы сервисного центра и обеспечивает гибкую, масштабируемую и удобную структуру для хранения и обработки информации, необходимой для эффективной работы виртуального помощника и автоматизации обработки клиентских обращений.

2.5 Моделирование логики работы

Моделирование логики работы представляет собой процесс построения упрощённого и структурированного описания последовательности действий, правил и взаимосвязей, которые определяют поведение системы или её компонентов при выполнении конкретных задач. Цель моделирования логики — наглядно представить, как система обрабатывает входные данные, какие решения принимает и каким образом реагирует на различные события и ситуации.

Этот процесс позволяет разработчикам и заинтересованным сторонам понять внутренние механизмы функционирования системы, выявить ключевые этапы обработки информации и определить, каким образом происходит взаимодействие между различными элементами. В результате создаётся чёткая схема или модель, которая отражает последовательность операций, условия переходов и действия, необходимые для выполнения поставленных задач.

Моделирование логики особенно важно при проектировании сложных систем, где требуется автоматизация многих процессов, включая обработку запросов, принятие решений и распределение ресурсов. Оно помогает систематизировать работу, выявить потенциальные узкие места и избыточные операции, а также упростить последующую разработку и внедрение программных компонентов.

В контексте создания виртуального помощника моделирование логики работы позволяет определить, каким образом бот будет взаимодействовать с пользователями, какие сценарии общения он должен поддерживать, как обрабатывать различные типы запросов и в каких случаях передавать задачу человеку-оператору. Это обеспечивает эффективное и предсказуемое поведение системы, минимизируя ошибки и повышая качество обслуживания.

Таким образом, моделирование логики работы является фундаментальной частью проектирования информационных систем, позволяя перейти от абстрактных требований к конкретным и реализуемым алгоритмам, обеспечивая прозрачность и управляемость процессов внутри системы [20].

Моделирование логики работы системы при работе с клиентом 11.

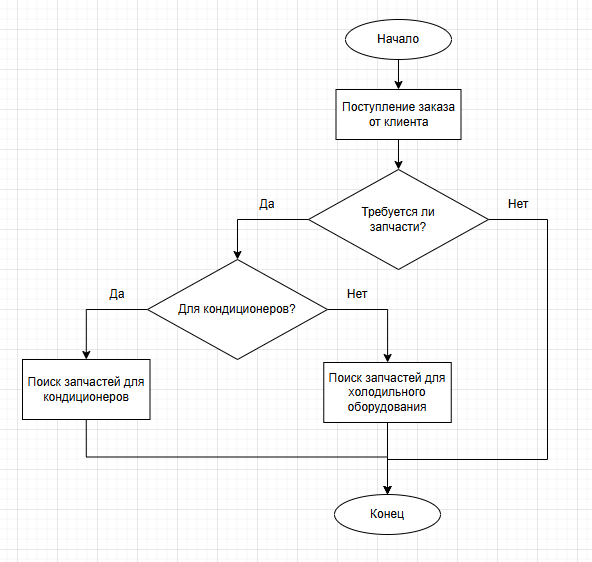


Рисунок 11 – Моделирование логики работы системы при работе с клиентом

В представленной блок-схеме описан процесс обработки заказа от клиента, который начинается с момента поступления заявки. Процесс структурирован таким образом, чтобы эффективно определить, требуется ли заказчику запчасти, и в зависимости от этого направить дальнейшие действия.

Основные этапы процесса:

1. Начало: Процесс начинается с поступления заказа от клиента, что является отправной точкой для дальнейших действий.
2. Поступление заказа от клиента - на данном этапе система фиксирует входящий заказ, что позволяет инициировать проверку необходимости запчастей.
3. «Требуются ли запчасти?» это ключевой вопрос, который определяет направление дальнейших действий. В зависимости от ответа «Да» или «Нет» процесс будет развиваться по разным ветвям.

Если ответ «Да», система уточняет, требуется ли запчасти для кондиционеров.

Если ответ «Да» на вопрос о кондиционерах, происходит поиск запчастей для кондиционеров. Это позволяет сосредоточиться на конкретных деталях, которые необходимы для выполнения заказа.

Если ответ «Нет», процесс переходит к поиску запчастей для холодильного оборудования, что обеспечивает гибкость в обслуживании различных типов оборудования.

Если ответ «Нет» на вопрос о запчастях, процесс завершается без дополнительных действий, переходя сразу к Концу.

4. Конец - процесс завершается, когда все необходимые действия выполнены или когда нет необходимости в запчастях.

Данная блок-схема демонстрирует четкую структуру процесса обработки заказов и облегчает оперативное принятие решений. Эффективная организация этапов позволяет минимизировать временные затраты и повысить качество обслуживания клиентов [5].

Моделирование логики работы системы при работе с заказчиком представлена на рисунке 12.

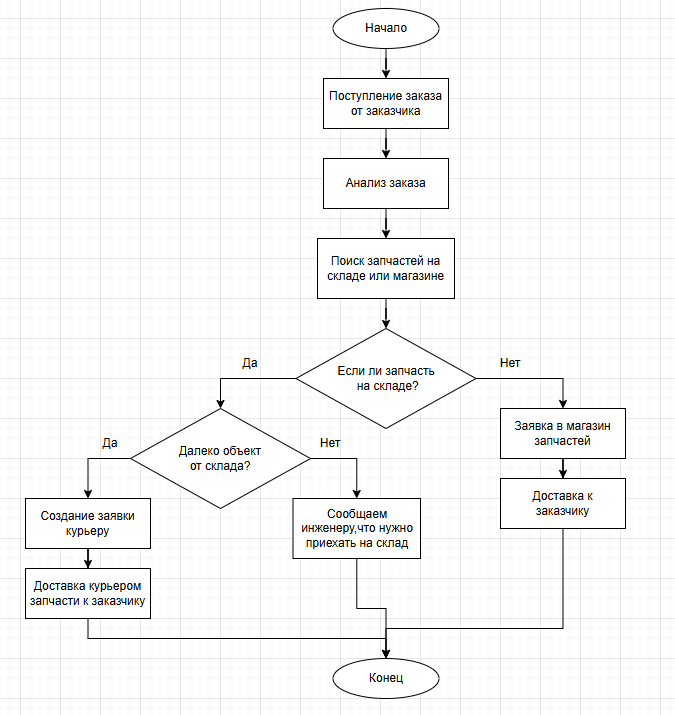


Рисунок 12 – Моделирование логики работы системы при работе с заказчиком

Представленная блок-схема иллюстрирует процесс обработки заказа запчастей от момента его поступления до завершения доставки. Этот процесс включает в себя несколько ключевых этапов, направленных на обеспечение эффективного и своевременного обслуживания клиентов.

Основные этапы процесса:

1. Начало - процесс начинается с инициализации обработки заказа.

2. Поступление заказа от заказчика - на этом этапе система фиксирует входящий заказ, что является отправной точкой для дальнейшего анализа.

3. Анализ заказа специалисты проверяют детали заказа, чтобы определить необходимые действия.

4. Поиск запчастей на складе или в магазине - система инициирует поиск необходимых запчастей, основываясь на анализе заказа.

5. «Если ли запчасть на складе?» - на этом этапе принимается ключевое решение о наличии запчастей.

6. Конец: Процесс завершается после успешной доставки запчастей, обеспечивая тем самым выполнение заказа.

Эта блок-схема демонстрирует четкую организацию процесса обработки заказов, что позволяет оптимизировать взаимодействие между всеми участниками. Структурированный подход обеспечивает быстрое решение вопросов, связанных с наличием запчастей, и позволяет эффективно управлять доставкой, что в конечном итоге повышает уровень удовлетворенности клиентов.

В ходе проектирования была проведена комплексная работа по анализу предметной области и организационной структуры компании, что позволило чётко определить ключевые требования и особенности функционирования сервисного центра. Были разработаны модели данных и логики работы системы, обеспечивающие эффективное взаимодействие между пользователями, заказами и запчастями. Создана архитектура программного виртуального помощника, способствующая автоматизации обработки клиентских обращений и оптимизации рабочих процессов. Все эти этапы заложили прочную основу для последующей реализации и внедрения решения, направленного на повышение качества обслуживания и сокращение времени отклика [6].

3 РАЗРАБОТКА СИСТЕМЫ

3.1 Этапы разработки

Этапы разработки — это последовательность логически связанных шагов, которые проходят при создании программного обеспечения, начиная от формулировки идеи и заканчивая внедрением готовой системы в рабочую среду. Каждый этап играет важную роль и обеспечивает поэтапное приближение к конечному результату, позволяя контролировать качество, выявлять и устранять ошибки, а также гибко адаптироваться к изменениям требований или условий.

На самых ранних стадиях осуществляется сбор и анализ требований. Это позволяет понять, что именно должно выполнять разрабатываемое программное обеспечение, какие проблемы решать и какие задачи автоматизировать. Затем начинается этап проектирования, в рамках которого создаётся архитектура системы, продумывается структура базы данных, взаимодействие компонентов и моделируется логика работы. Это основа, на которой строится дальнейшая реализация.

Следующий этап — это собственно программирование, или реализация. Здесь разработчики создают код, используя выбранные технологии и инструменты, формируют пользовательский интерфейс и обеспечивают интеграцию всех элементов в единую рабочую систему. Особое внимание уделяется корректности, безопасности и надёжности создаваемых решений.

После этого наступает фаза тестирования, когда проводится проверка работоспособности всех функций, поиск возможных ошибок, проверка соответствия требованиям. Тестирование помогает выявить недочёты и убедиться, что система работает стабильно в различных ситуациях.

На завершающем этапе осуществляется внедрение, то есть установка и настройка программного продукта в рабочей среде предприятия. Также может выполняться обучение пользователей, сопровождение запуска и техническая поддержка. Иногда после внедрения проводится этап сопровождения и развития, когда в систему добавляются новые функции или адаптация под меняющиеся условия.

Таким образом, этапы разработки позволяют структурировать весь процесс создания программного продукта, обеспечивают прозрачность, управляемость и возможность итерационного совершенствования, что особенно важно при создании сложных и востребованных решений, таких как виртуальный помощник для сервисного центра.

Разработка виртуального помощника велась с применением спиральной модели, которая сочетает итеративный подход с элементами анализа рисков и постепенного наращивания функциональности. Каждый виток спирали включал в себя планирование, моделирование, реализацию и проверку, что позволило гибко адаптироваться к требованиям и техническим ограничениям в процессе создания системы [11].

На первом этапе был выполнен предварительный анализ предметной области и возможностей интеграции. Определены ключевые компоненты системы: интерфейс взаимодействия с пользователем через *Telegram*, связующее звено в виде визуальной среды автоматизации *n*8*n*, а также внешняя интеллектуальная система — *API* нейросети, отвечающая за генерацию ответов. Были проанализированы риски, связанные с доступностью *API*, стабильностью соединения с *Telegram* и возможностями масштабирования сценариев в *n*8*n*.

Затем был реализован базовый виток системы: с помощью сервиса *BotFather* в *Telegram* был создан бот и получен *API*-токен. Этот токен был использован в *n*8*n* для настройки начального соединения между платформой и ботом. Далее был добавлен блок *Telegram* *Trigger*, позволяющий принимать входящие сообщения от пользователей. Это стало основой для всей дальнейшей логики.

На следующем витке спирали была разработана схема маршрутов обработки сообщений. Полученные от клиента данные направлялись через промежуточные блоки на *API* нейросети, где происходила интеллектуальная генерация ответа. После получения ответа *n*8*n* автоматически возвращал его в чат *Telegram*-бота, формируя ощущение живого диалога. В процессе был произведён контроль корректности данных, структуры сообщений и безопасности обмена.

Следующим этапом стало постепенное расширение функциональности: добавление условий ветвления логики, создание шаблонов ответов на типовые запросы, внедрение переменных, отслеживание статуса обращений, а также реализация первичных механизмов логирования и уведомлений для администратора в случае ошибок. Параллельно проводилось тестирование каждого блока на устойчивость и корректность работы с нестандартными или некорректными сообщениями.

С каждым новым витком спирали структура сценариев усложнялась, появлялись дополнительные логические условия, а также была проведена оптимизация обработки повторяющихся запросов. Это позволило достичь большей гибкости и повысить отзывчивость системы. Платформа *n*8*n* обеспечивала удобную визуализацию потоков данных, что упрощало анализ и устранение ошибок, а также ускоряло процесс доработки.

На финальных этапах был проведён анализ производительности, протестированы предельные нагрузки, а также проверена устойчивость к обрывам связи с внешними сервисами. Кроме того, была реализована базовая логика отказоустойчивости — например, повторная отправка запроса в случае отсутствия ответа от ИИ или автоматическое уведомление администратора при превышении времени обработки.

В результате применения спиральной модели удалось создать работоспособную, расширяемую систему, способную эффективно взаимодействовать с пользователем в реальном времени через *Telegram*, обращаться к внешнему интеллектуальному модулю и динамически реагировать на входящие обращения с минимальной задержкой. Такой подход обеспечил высокую гибкость разработки, быструю проверку гипотез и устойчивость архитектуры при наращивании функциональности [8].

3.2 Средства разработки

Для реализации проекта виртуального помощника были выбраны современные и удобные инструменты разработки, каждый из которых обеспечил отдельный уровень в архитектуре системы — от интерфейса общения с пользователем до логики обработки и подключения искусственного интеллекта. Ниже приведён перечень основных средств разработки с их подробным описанием и обоснованием выбора:

1. *Telegram* (*Bot* *API*)

*Telegram* выступает в качестве основной платформы взаимодействия между пользователем и системой.  
Причины выбора:

1. Широкое распространение мессенджера и доступность *API*;
2. Поддержка текстового ввода, кнопок и вложений;
3. Простота регистрации и подключения бота через сервис *BotFather*;
4. Возможность мгновенной доставки сообщений без задержек.

*Telegram*-бот создаёт ощущение общения с реальным консультантом, не требует установки отдельного приложения и обеспечивает высокий уровень удобства для пользователя [1].

1. *n*8*n* (*Workflow* *Automation* *Platform*)

*n*8*n* используется как центральный оркестратор, обрабатывающий сценарии взаимодействия, направляющий запросы и получающие ответы.  
Преимущества использования:

1. Поддержка *no*-*code*/*low*-*code* подхода: визуальный редактор сценариев;
2. Более 300 интеграций: *Telegram*, *Webhooks*, *HTTP*-запросы, *Google* *Sheets* и др;
3. Возможность условной логики, циклов, ветвлений и переменных;
4. Открытый исходный код (*open*-*source*), бесплатное локальное размещение.

С помощью *n*8*n* удалось быстро построить цепочки автоматической обработки: от получения сообщения до отправки ответа пользователю. Интерфейс позволяет без программирования подключать *API* и строить гибкие алгоритмы [7].

1. *OpenAI* *API* (или другая ИИ-система)

Для генерации осмысленных и полезных ответов на пользовательские сообщения применялся *API* языковой модели.  
Причины интеграции:

1. Поддержка естественного языка (*NLP*);
2. Возможность настройки контекста общения;
3. Гибкость и масштабируемость;
4. Высокое качество генерации текста.

ИИ-система анализирует текст обращения пользователя, формирует релевантный ответ и возвращает его в цепочку обработки *n*8*n* [2].

1. *JavaScript* / *JSON* (внутри *n*8*n*)

Несмотря на визуальный подход, в *n*8*n* активно используется встроенный *JavaScript* и работа с *JSON*-данными.  
Функции:

1. Создание пользовательских скриптов (например, очистка текста, обработка ошибок);
2. Работа с переменными, логикой и условиями;
3. Генерация или парсинг структурированных данных между узлами.
4. Сервер *Docker* (или локальный *Node*.*js*)

Для запуска платформы *n*8*n* использовался локальный сервер, часто развёрнутый через *Docker* [7].  
Преимущества:

1. Быстрая установка и масштабируемость;
2. Независимость от ОС;
3. Возможность автоматического запуска при старте системы;
4. Легкое развертывание на *VPS* или в облаке.

Дополнительные инструменты:

1. *Postman* — тестирование *API*-запросов к внешним сервисам;
2. *VS* *Code* — при необходимости редактирования *JavaScript*-функций;
3. *Google* *Sheets* — в качестве простого хранилища заявок и логов;
4. *Telegram* *Webhook* или *Polling* — варианты подключения бота к платформе.

Использование этих инструментов в совокупности обеспечило быстрый запуск, удобную настройку, визуальную наглядность логики и возможность масштабирования проекта. Такой стек подходит как для прототипа, так и для полноценного решения в условиях реального сервисного центра [8].

3.3 Интерфейс пользователя

Пользовательский интерфейс (*UI*) — это точка взаимодействия между человеком и системой (программой, устройством или сервисом). В контексте автоматизации сервисного центра под интерфейсом подразумевается:

1. Способы ввода информации: текстовые сообщения, голосовые команды, нажатие кнопок.
2. Способы вывода информации: текстовые ответы, изображения, интерактивные меню.
3. Логика взаимодействия: сценарии диалога, обработка запросов, перенаправление данных.

Хороший интерфейс должен быть:

1. Интуитивным — пользователь понимает, как работать с системой без инструкций.
2. Доступным — поддерживает разные формы взаимодействия (текст, аудио, кнопки).
3. Эффективным — минимизирует время на выполнение задач (например, оформление заявки).

Для сервисного центра критично, чтобы интерфейс работал на устройствах клиентов без установки дополнительного ПО и был привычным [10].

*Telegram* — оптимальная платформа для автоматизации сервисного центра благодаря следующим преимуществам:

1. Нулевой порог входа. Клиентам не нужно скачивать отдельное приложение — *Telegram* уже установлен у 80% пользователей.
2. Знакомый интерфейс. Большинство умеет пользоваться чатами, что снижает нагрузку на обучение.
3. Мультиформатность. Поддержка текста, голосовых сообщений, фото и документов (например, можно отправить фото чека или серийного номера устройства).
4. Кроссплатформенность. Доступ с телефона (*iOS*/*Android*), компьютера или через веб-версию.
5. Готовая экосистема. *Telegram* предоставляет *API* для создания ботов с богатым функционалом:
   1. Интерактивные кнопки и меню.
   2. Автоматическая обработка данных (например, извлечение номера заказа из сообщения).
   3. Интеграция с *CRM* через вебхуки.
6. Экономия ресурсов. Не требуется разрабатывать и поддерживать отдельное приложение.
7. Безопасность. Возможность настройки доступа (например, только для авторизованных клиентов) и шифрование данных.

Примеры эффективного использования:

1. Оформление заявок. Клиент выбирает услугу из меню бот запрашивает недостающие данные автоматически создает заявку в *CRM*.
2. Отслеживание статуса. По номеру заказа бот выводит актуальную информацию из базы данных.
3. Обратная связь. После завершения ремонта бот отправляет опрос удовлетворенности.

*Telegram*-бот — это универсальный интерфейс, который:

1. Снижает нагрузку на операторов за счет автоматизации рутинных запросов.
2. Ускоряет обслуживание клиентов (среднее время оформления заявки сокращается с 10 минут до 30 секунд).
3. Работает в привычной для пользователей среде без дополнительных затрат на разработку.

Для сервисного центра, где важны оперативность и удобство, *Telegram* становится идеальным решением, обеспечивая бесшовное взаимодействие между клиентом и бизнесом [3].

Интерфейс пользователя с поиском запчасти в интернете представлен на рисунке 13.

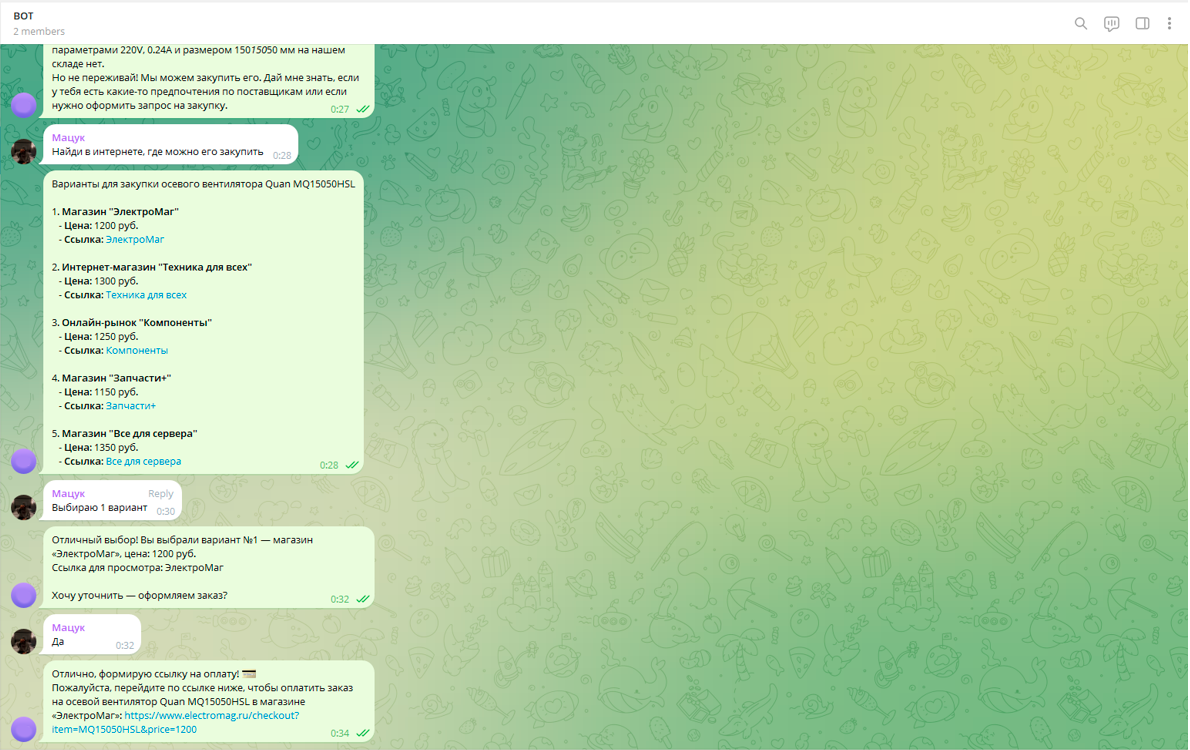


Рисунок 13 – Интерфейс пользователя с поиском запчасти в интернете

В рассматриваемом сценарии виртуальный помощник выполнял функцию ассистента сервисного центра в ситуации, когда нужной запчасти не оказалось на складе. Пользователь — сотрудник или клиент — через интерфейс *Telegram*-бота ввёл запрос на осевой вентилятор *Quan* *MQ*15050*HSL*. В ответ система проверила наличие данной позиции на внутреннем складе и, не обнаружив её, автоматически переключилась на альтернативный сценарий — поиск вариантов закупки в интернет-магазинах.

Бот сформировал и выдал пользователю список из пяти доступных предложений от различных онлайн-продавцов. Каждое предложение включало название магазина, цену и активную ссылку на страницу товара. После ознакомления с вариантами пользователь выбрал подходящий — первый магазин «ЭлектроМаг» с ценой 1200 рублей.

На следующем этапе виртуальный помощник уточнил у пользователя, желает ли он оформить заказ. После подтверждения бот сформировал ссылку на оформление и оплату, тем самым обеспечив удобный и быстрый переход к покупке, без необходимости вручную искать и сравнивать предложения.

Данный сценарий демонстрирует адаптивную логику работы системы: при отсутствии запчасти в локальной базе данных она переходит к автоматизированному поиску в интернете, тем самым сокращая время на решение проблемы и повышая общую оперативность обслуживания.

Интерфейс пользователя с поиском запчасти на складе представлен на рисунке 14.

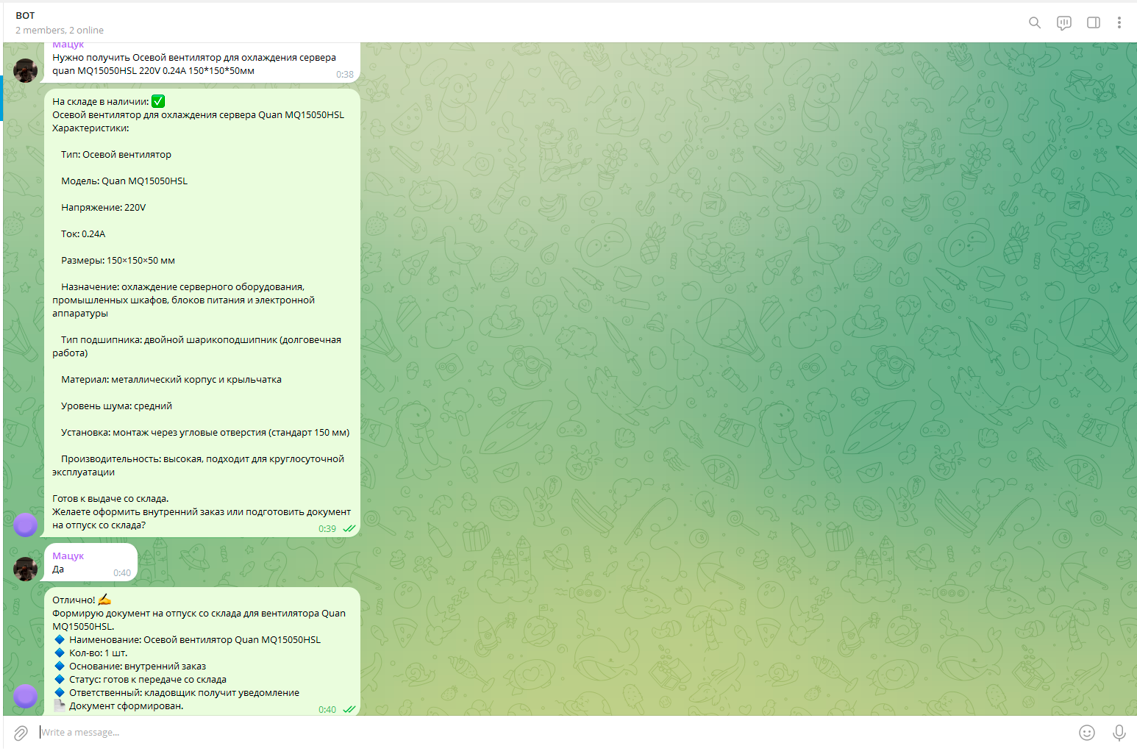


Рисунок 14 – Интерфейс пользователя с поиском запчасти на складе

Пользователь отправляет в *Telegram*-бота сообщение с запросом на определённую запчасть, например:  
«Нужен вентилятор для охлаждения *Quan* *MQ*15050*HSL*».

Бот получает текст запроса и автоматически обрабатывает его. На первом этапе он выделяет ключевые параметры — наименование, модель, тип устройства, габариты и другие технические характеристики, если они указаны. Далее запускается механизм проверки наличия запрошенной позиции на складе организации.

Система подключается к внутреннему справочнику или базе данных склада, где хранится информация о текущих остатках товаров, включая артикулы, названия, количество, технические параметры, а также информацию о локации (полке, ячейке и пр.).

Если запрашиваемая деталь находится на складе, бот формирует сообщение с подтверждением наличия. В этом сообщении он указывает:

1. Точное наименование позиции,
2. Её характеристики (модель, напряжение, размер, назначение и т.д.),
3. Статус «в наличии»,
4. Предложение оформить внутренний заказ или подготовить документ на отпуск со склада.

После подтверждения от пользователя, что деталь действительно нужна, бот передаёт данные в 1С или другую систему учёта, создавая заявку на отпуск со склада. Одновременно может быть сформирован внутренний документ (например, форма ТОРГ-29) с указанием ответственного лица и номера заказа. Также бот уведомляет кладовщика или другого ответственного сотрудника о необходимости подготовки товара к выдаче.

Финальным этапом является отправка пользователю уведомления о том, что товар готов к получению или уже передан в соответствующее подразделение.

Интерфейс пользователя с поиском запчасти и отработкой исключения представлен на рисунке 15.

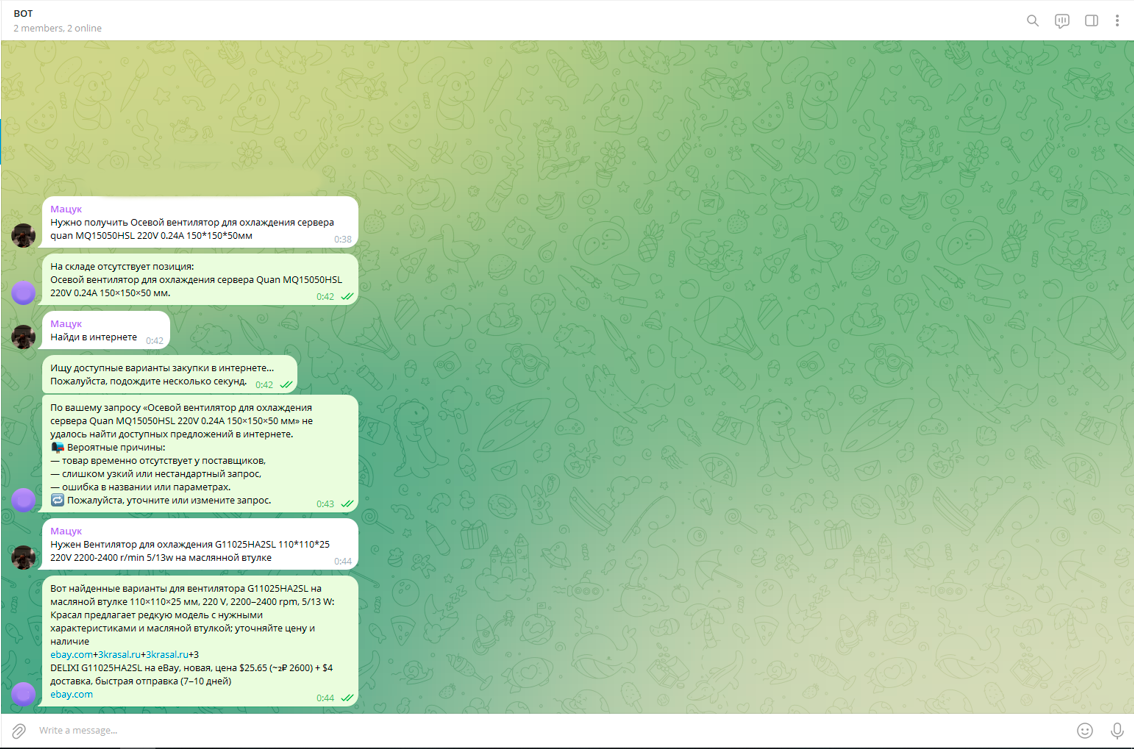


Рисунок 15 – Интерфейс пользователя с поиском запчасти и отработкой исключения

Пользователь обращается к *Telegram*-боту с сообщением:  
«Нужен вентилятор для охлаждения *Quan* *MQ*15050*HSL* 220*V* 0.24*A* 150×150×50 мм».

Бот принимает сообщение, выполняет анализ текста и извлекает ключевые параметры: наименование устройства, модель, характеристики. После этого система выполняет запрос к внутренней базе данных склада для поиска соответствующей детали.

На этом этапе может произойти одно из двух:

1. Деталь не найдена вообще — ни точного совпадения по названию, ни частичных совпадений по параметрам.
2. Найдены только похожие позиции, но не в полной мере соответствующие запросу (например, другая модель или характеристики).

Если система не может уверенно определить нужную позицию, бот информирует пользователя:  
«К сожалению, нужная деталь не найдена на складе. Возможно, указаны слишком точные или нестандартные параметры. Пожалуйста, уточните модель, размер или назначение детали. Например, попробуйте сократить запрос до “вентилятор 150×150×50 220*V*».

После этого пользователь может:

1. Скорректировать запрос, указав более общие или альтернативные данные,
2. Воспользоваться подсказками от бота (например, предложенными аналогами или популярными моделями),
3. Запросить помощь специалиста (если бот интегрирован с оператором).

Как только пользователь отправляет обновлённый или уточнённый запрос, процесс запускается повторно. Бот снова анализирует сообщение и делает новый поиск на складе. Если теперь позиция найдена — запускается обычный сценарий обработки заказа. Если снова не найдена, бот предлагает перейти к поиску в интернет-магазинах.

3.6 Серверная часть системы

Серверная часть проекта представляет собой ключевое звено архитектуры, обеспечивающее обработку запросов, маршрутизацию данных, выполнение логики и интеграцию с интеллектуальной системой генерации ответов. Основной компонент серверной части — это установленная и настроенная система *n*8*n*, которая выполняет функции визуального оркестратора и служит связующим элементом между интерфейсом общения (*Telegram*) и внешним ИИ-сервисом (например, *OpenAI*).

Установка *n*8*n* производится либо локально, либо на облачном сервере с помощью *Docker*-контейнера или *Node*.*js*. Такой подход обеспечивает гибкость в развёртывании, простоту в управлении и независимость от платформы. После установки производится настройка окружения: указывается порт работы, параметры безопасности и конфигурация переменных окружения (например, токены для *Telegram* и *API*-ключ для ИИ).

Далее осуществляется подключение *Telegram*-бота через специальный блок *Telegram* *Trigger*, который реагирует на входящие сообщения. Этот триггер запускает цепочку логики в *n*8*n*, внутри которой сообщение обрабатывается, очищается от лишних символов (при необходимости), и отправляется на внешний *API* с помощью блока *HTTP* *Request*. Запрос формируется в формате *JSON* и содержит текст сообщения, роль бота, дополнительные параметры диалога и ограничения по длине ответа. Полученный от нейросети результат обрабатывается и направляется обратно пользователю через *Telegram* *Send* *Message*.

Также в серверной части может использоваться логика сохранения обращений, фильтрация с автоматической классификации обращений и условий обработки на основе содержимого. Платформа *n*8*n* позволяет добавлять блоки проверки ошибок, логировать обращения в таблицы *Google* *Sheets* или базы данных, а также уведомлять администратора в случае нештатных ситуаций.

Таким образом, серверная часть, построенная на базе *n*8*n*, становится универсальной, визуально понятной и легко настраиваемой платформой, обеспечивающей непрерывную работу виртуального помощника. Интеграция с внешним ИИ значительно расширяет функциональность, позволяя системе давать осмысленные, контекстно-зависимые ответы и обрабатывать широкий спектр клиентских запросов без участия оператора [9].

Описание серверной структуры системы представлено на рисунке 16.

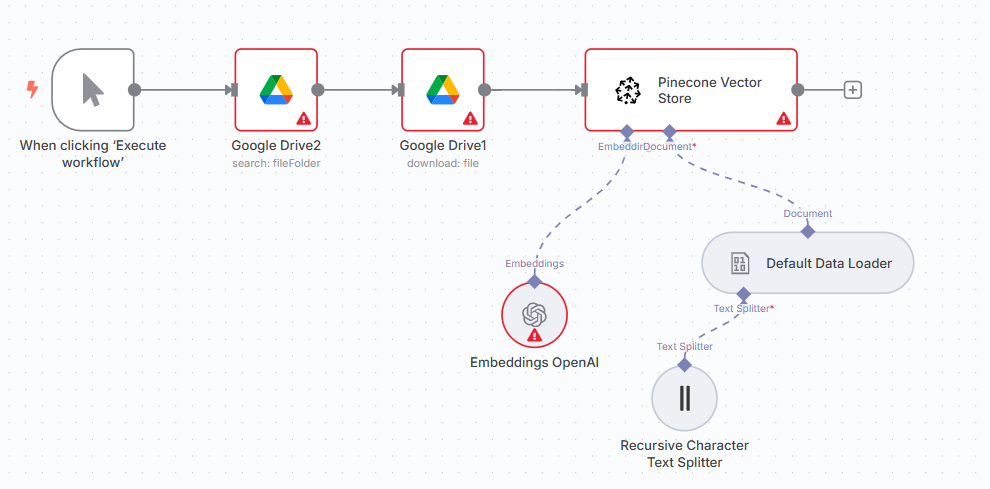


Рисунок 16 – Описание серверной структуры системы

На диаграмме представлен последовательный процесс выполнения рабочего процесса, состоящий из нескольких ключевых этапов, каждый из которых выполняет свою уникальную функцию.

1. Первый блок: «Когда нажата кнопка 'Выполнить рабочий процесс»

Этот блок служит триггером для начала всего процесса. Пользователь инициирует рабочий процесс нажатием кнопки, что запускает последовательность действий.

1. Второй блок: «*Google* *Drive*2», «поиск: *fileFolder*»

После инициации рабочего процесса осуществляется поиск необходимой папки или файла в *Google* *Drive*2. Этот этап включает в себя взаимодействие с облачным хранилищем для доступа к нужным данным.

1. Третий блок: «*Google* *Drive*1», «скачать: файл»

На этом этапе, после успешного поиска, система скачивает файл из *Google* *Drive*1. Это важный шаг, так как файл будет использоваться в дальнейшем процессе.

1. Четвертый блок: «*Pinecone* *Vector* *Store*», «*EmbedDocument*»

Загруженный файл затем передается в *Pinecone* *Vector* *Store*, где происходит его обработка и встраивание (*embedding*). Этот этап необходим для того, чтобы подготовить данные к дальнейшей обработке и анализу.

1. Пятый блок: «*Default* *Data* *Loader*»

Далее данные загружаются с использованием стандартного загрузчика данных. Этот блок отвечает за интеграцию данных в систему, что позволяет обеспечить их доступность для последующих этапов.

1. Шестой блок: «*Embeddings* *OpenAI*»

На этом этапе происходит использование модели *OpenAI* для создания встраиваний (*embeddings*), что позволяет улучшить качество обработки текстовой информации. Этот шаг критически важен для дальнейшего анализа данных.

1. Седьмой блок: «Рекурсивный делитель текста»

Заключительный этап включает использование рекурсивного делителя текста, который обрабатывает и структурирует текстовые данные для удобства дальнейшего анализа и использования.

Данная блок-схема иллюстрирует четкую последовательность действий в рамках рабочего процесса, начиная с инициации и заканчивая обработкой данных. Каждый блок выполняет свою функцию, обеспечивая эффективное взаимодействие между различными системами и компонентами [20].

Логика взаимодействия серверных компонентов представлена на рисунке 17.

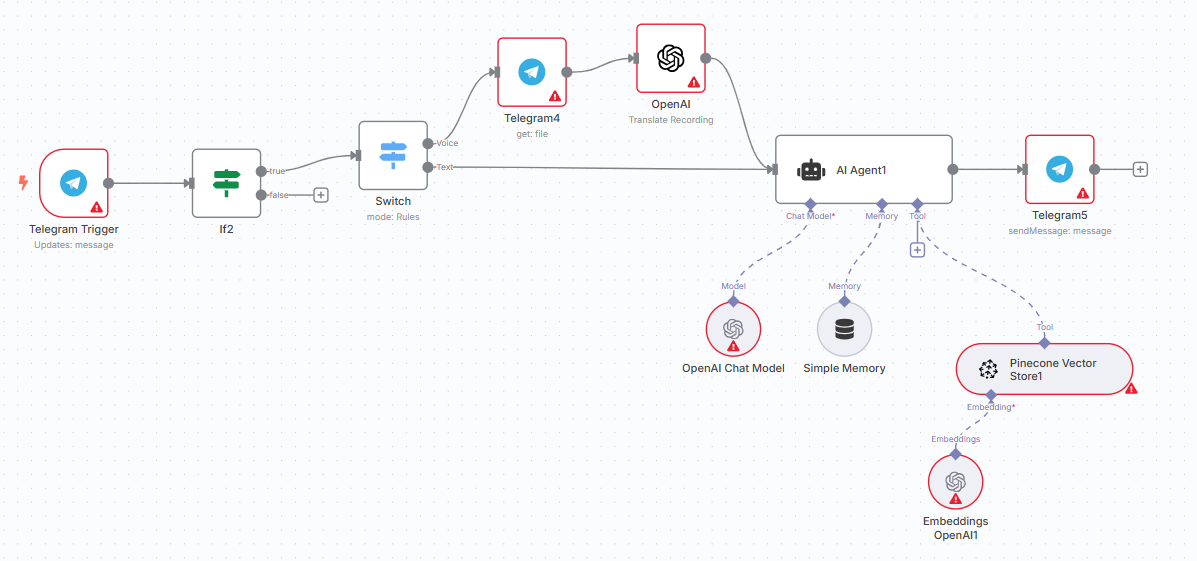


Рисунок 17 – Логика взаимодействия серверных компонентов

На представленной диаграмме изображен последовательный рабочий процесс, который, применяется в визуальной среде программирования для создания приложений на базе искусственного интеллекта. Процесс развивается слева направо и включает в себя несколько ключевых элементов и ветвлений.

Основные элементы процесса:

1. *Telegram* *Trigger*: *Updates*: *message*

Этот элемент служит стартовой точкой для рабочего процесса, активируясь при получении сообщения в *Telegram*.

1. *If*2

Условный блок, который определяет, какие действия необходимо предпринять в зависимости от входных данных.

1. *Switch*: *mode*: *Rules*

Блок маршрутизации, который направляет поток в зависимости от заранее определенных правил. Он имеет два выхода: «*Voice*» и «*Text*», что позволяет обрабатывать входные данные в зависимости от их типа.

1. *Telegram*4: *get* *file*

Этот элемент связан с получением файла из *Telegram*, что особенно актуально в случае, если входные данные представляют собой голосовое сообщение.

1. *OpenAI*: *Translate* *Recording*

Здесь используется модель *OpenAI* для перевода аудиозаписи, что позволяет преобразовать голосовые сообщения в текст.

1. *AI* *Agent*!

Основной компонент *AI*-агента. Он принимает три типа входных данных.

1. *OpenAI* *Chat* *Model*: *Model*

Этот элемент, вероятно, использует языковую модель *OpenAI* для обработки текстовой информации.

1. *Simple* *Memory*: *Memory*

Блок, предназначенный для краткосрочной памяти, что помогает *AI*-агенту сохранять контекст взаимодействия.

1. *Pinecone* *Vector* *Store*1: *Tool*

Векторная база данных, используемая для семантического поиска, принимает векторные встраивания (*embeddings*).

1. *Embeddings* *OpenAI*: *Embedding*

Элемент, который генерирует векторные встраивания для базы данных *Pinecone*, позволяя осуществлять более точный поиск информации.

1. *Telegram*5: *sendMessage*: *message*

Заключительный элемент, отправляющий сообщение через *Telegram*, завершая рабочий процесс.

Связи и символы.

Элементы схемы соединены линиями, указывающими на поток данных и управление. Линии с пунктиром обычно обозначают дополнительные связи или типы данных, передаваемые между элементами.

Кроме того, на схеме присутствуют символы предупреждений (красные треугольники), указывающие на возможные ошибки или предупреждения.

Эта диаграмма представляет собой структурированное отображение рабочего процесса, демонстрируя, как данные обрабатываются и передаются между различными компонентами системы [14].

3.7 Входные данные и выходные данные

Входные и выходные данные — это фундаментальные понятия, лежащие в основе работы любой информационной системы. Они описывают процесс получения информации системой, её обработки и предоставления результата пользователю или другой системе.

Входные данные — это вся информация, которая поступает в систему извне и служит исходным материалом для обработки. Эти данные могут быть получены от пользователя (например, текст, введённый в мессенджере), с других программных платформ (через *API*), из базы данных, файлов или датчиков. В контексте программных решений, таких как виртуальные помощники, входными данными чаще всего являются текстовые сообщения, команды, параметры выбора, идентификаторы пользователей и прочие данные, необходимые для выполнения запроса.

Выходные данные — это результат, который формирует система после обработки входной информации. Это может быть текстовый ответ, файл, изображение, отчёт, ссылка, результат вычислений, изменение статуса заказа, либо другой структурированный или неструктурированный ресурс. Выходные данные возвращаются пользователю или передаются во внешнюю систему для дальнейших действий.

Проще говоря, система принимает информацию, обрабатывает её по определённому алгоритму или логике, и выдаёт результат. Это взаимодействие лежит в основе всех программных решений — от простых калькуляторов до сложных автоматизированных систем с искусственным интеллектом. Важно, чтобы входные данные были корректными и понятными для системы, а выходные — полезными, своевременными и соответствующими ожиданиям пользователя [9].

В рамках разработки интеллектуального виртуального помощника, реализованного на базе *Telegram*-бота, особое значение приобретают процессы обработки входных и выходных данных, которые определяют логику взаимодействия пользователя с системой и эффективность выполнения задач, связанных с обслуживанием клиентов сервисного центра.

Входными данными выступает текстовая информация, которую пользователь вводит в интерфейсе мессенджера *Telegram*. Это может быть свободный текст, содержащий описание проблемы, запрос на определённую услугу, просьба уточнить наличие запчастей, а также указание на модель устройства, дату возможного визита или номер уже оформленного заказа. Пользователь может писать сообщения в произвольной форме, как в формате деловой переписки, так и в виде краткой фразы — именно поэтому система должна уметь обрабатывать неструктурированный текст, извлекать из него ключевую информацию и передавать в обработку через серверную логику.

Полученные входные сообщения поступают в платформу обработки (в данном случае — *n*8*n*), где они проходят этапы фильтрации, анализа и маршрутизации. При необходимости, текст дополнительно анализируется при помощи ИИ, способного интерпретировать намерения пользователя и классифицировать запрос по типу: консультация, оформление заказа, проверка наличия запчастей, получение документа или другое. Благодаря этому виртуальный помощник может реагировать не просто шаблонно, а с учётом контекста и конкретной ситуации.

Выходные данные формируются как результат реакции системы на входящие запросы. Они представляют собой конкретные информационные и файловые ресурсы, которые отправляются обратно пользователю через *Telegram*-бота. В зависимости от типа запроса, это могут быть:

1. Файлы с описанием нужных запчастей, включающие спецификации, цены и возможные аналоги;
2. Ссылки на оплату, автоматически сгенерированные с привязкой к конкретному заказу и интегрированные с платёжной системой;
3. Электронные чеки и квитанции, подтверждающие приём оплаты или факт обращения в сервис;
4. Ссылки на трекинг доставки или статус выполнения заявки;
5. Карта ближайших пунктов обслуживания, если предполагается визит клиента в сервисный центр;
6. Служебные сообщения с подтверждением действий, напоминаниями и уточняющими вопросами.

Таким образом, вся система построена на принципе: пользователь вводит текстовый запрос в привычной для себя форме, а взамен получает релевантный, структурированный и, при необходимости, документально подтверждённый ответ. Такая логика взаимодействия обеспечивает максимальное удобство, оперативность и автоматизацию процессов, ранее требовавших участия оператора [6].

3.8 Тестирование системы

Тестирование системы — это важнейший этап жизненного цикла разработки программного обеспечения, направленный на проверку корректности, надёжности и соответствия работы системы заданным требованиям. Его основная цель — выявить ошибки, недочёты и несоответствия ещё до того, как программный продукт будет внедрён или передан конечному пользователю.

В процессе тестирования проводится всесторонняя проверка работы системы в различных условиях, с разными входными данными, а также в граничных и нестандартных ситуациях. Тестированию подвергаются как отдельные компоненты (например, модули, функции, блоки логики), так и вся система в целом, включая её взаимодействие с внешними сервисами, базами данных, *API*, интерфейсом пользователя и другими элементами инфраструктуры.

Тестирование позволяет убедиться, что система:

1. Выполняет свои функции так, как это предусмотрено в техническом задании;
2. Обрабатывает данные корректно, независимо от сложности или характера ввода;
3. Устойчива к ошибкам и нештатным ситуациям;
4. Демонстрирует приемлемую производительность и стабильность при повышенной нагрузке;
5. Надёжно сохраняет и передаёт информацию, не нарушая целостности данных.

В современных проектах может использоваться как ручное, так и автоматизированное тестирование. Ручное предполагает участие человека, который последовательно проверяет работу системы. Автоматизированное — это использование специальных скриптов и инструментов, которые автоматически выполняют тесты, сравнивают фактический результат с ожидаемым и формируют отчёты.

Тестирование — это не просто поиск ошибок. Это процесс обеспечения качества, позволяющий минимизировать риски, повысить доверие к системе и обеспечить пользователю надёжный, удобный и предсказуемый в работе продукт. Без тестирования даже хорошо спроектированная система может вести себя нестабильно, давать сбои или не соответствовать ожиданиям пользователей.

В процессе разработки нашего программного решения для автоматизации обработки заявок в сервисном центре была проведена комплексная проверка работоспособности всех основных компонентов системы. Тестирование проводилось поэтапно, с фокусом на реальные сценарии взаимодействия пользователя с виртуальным помощником через *Telegram*, обработку сообщений на серверной части (*n*8*n*), корректность связи с внешним ИИ, а также генерацию выходных данных, таких как ссылки на оплату, сведения о запчастях и электронные документы [1].

В первую очередь была протестирована стабильность приёма и отправки сообщений через *Telegram*. Проверяли, насколько корректно бот воспринимает текстовые сообщения, фиксирует идентификаторы пользователей и передаёт их на обработку. Уделялось внимание как простым командам («Здравствуйте», «Нужна диагностика»), так и сложным обращениям с перечислением нескольких задач или нестандартных формулировок.

На втором этапе проверялась логика обработки в *n*8*n* — каждое входящее сообщение должно корректно запускать сценарий, включающий обработку текста, передачу его в ИИ и формирование ответа. Мы проверяли работу с пустыми сообщениями, некорректными командами, повторными запросами, а также ситуации, когда пользователь меняет тему диалога на ходу. Особое внимание уделялось тому, чтобы в этих случаях бот не «зависал» и корректно возвращал либо ответ, либо запрос на уточнение.

Также мы протестировали интеграцию с ИИ, обеспечивающую генерацию содержательных и осмысленных ответов. Проверялась точность, релевантность и полнота выдачи в ответ на типичные клиентские запросы, такие как «Сколько стоит замена блока питания?» или «Есть ли в наличии запчасть для модели *X*?». Результаты, полученные от ИИ, сопоставлялись с вручную составленными эталонными ответами.

Далее была проведена проверка формирования выходных данных, включая:

1. Генерацию файлов с описанием запчастей;
2. Формирование ссылок на оплату;
3. Отправку электронных чеков и уведомлений;
4. Корректную работу сценариев при подключении к базе данных и *api*.

На завершающем этапе тестировались краевые случаи — например, отсутствие связи с внешними сервисами, нестабильный интернет у клиента, повторная отправка сообщения, выход за лимит символов, ввод нераспознанных символов и другие непредвиденные ситуации.

По результатам тестирования система показала устойчивую работу, правильно обрабатывала разнообразные запросы, своевременно формировала нужные данные и демонстрировала высокую степень готовности к реальному применению в сервисном центре.

3.9 Экономическая эффективность разработки системы

Разработка и внедрение программного решения, в частности виртуального помощника для обработки клиентских обращений в сервисном центре, представляет собой инвестицию, направленную на повышение эффективности бизнес-процессов и снижение операционных затрат. Экономическая эффективность такого проекта проявляется не только в прямом снижении затрат на обслуживание, но и в более широком спектре выгод, затрагивающих всю структуру взаимодействия с клиентами, внутреннюю организацию труда и конкурентные преимущества компании.

Во-первых, автоматизация приёма и обработки заявок позволяет значительно сократить нагрузку на персонал, особенно в части рутинных задач. Виртуальный помощник способен круглосуточно принимать сообщения, отвечать на типовые вопросы, формировать заявки и направлять их в нужные подразделения без участия оператора. Это приводит к уменьшению потребности в дополнительном персонале, особенно в периоды пиковых нагрузок, а также сокращает время отклика на запросы клиентов, что в свою очередь способствует росту уровня удовлетворённости и лояльности клиентов.

Во-вторых, использование автоматизированной системы позволяет сократить количество ошибок, связанных с человеческим фактором. Точная фиксация данных, автоматическая генерация ответов и документов, строгая логика обработки заявок обеспечивают более высокую надёжность и прозрачность процессов. Это минимизирует риск потери клиентов из-за недоразумений, недосказанностей или забытых обращений.

Также важным аспектом является повышение производительности труда. Один оператор, ранее обрабатывавший ограниченное число заявок вручную, теперь может сосредоточиться на более сложных, нестандартных случаях, в то время как типовые обращения обрабатываются автоматически. Это способствует оптимальному использованию ресурсов и высвобождению времени для решения задач более высокого уровня.

Нельзя не отметить и снижение затрат на бумажные носители, телефонную связь и другие традиционные каналы коммуникации. Внедрение цифровых инструментов также открывает возможности для интеграции с платёжными системами, аналитическими сервисами и *CRM*-платформами, что позволяет компании более гибко управлять своей деятельностью и быстрее адаптироваться к изменениям спроса.

Таким образом, экономическая эффективность разработки виртуального помощника выражается в сокращении затрат, повышении скорости и качества обслуживания, снижении вероятности ошибок, а также в росте общей конкурентоспособности предприятия на рынке. При сравнительно невысокой стоимости внедрения и поддержки такая система способна принести значимые и устойчивые результаты уже в краткосрочной перспективе [14].

Для оценки экономической эффективности разработанной системы используется показатель срока окупаемости инвестиций, который рассчитывается по формуле:

*T*=*ISm*−*CmT* = *\frac*{*I*}{*S\_m* - *C\_m*}*T*=*Sm*​−*Cm*​*I*​

где *TTT* — срок окупаемости в месяцах, *III* — сумма первоначальных инвестиций на разработку и внедрение системы, *SmS\_mSm*​ — ежемесячная экономия от внедрения системы, *CmC\_mCm*​ — ежемесячные текущие расходы на поддержку и обслуживание.

В нашем случае инвестиции составляют примерно 400 000 рублей, что включает затраты на разработку, интеграцию и тестирование. Ежемесячная экономия складывается из сокращения затрат на зарплату сотрудников за счёт автоматизации рутинных задач и уменьшения ошибок, что мы оцениваем в среднем в 55 000 рублей. Текущие расходы на поддержку системы составляют около 5 000 рублей в месяц.

Подставляя эти значения в формулу, получаем:

*T*=400 00055 000−5 000=400 00050 000=8 месяцев*T* = *\frac*{400*\*,000}{55*\*,000 - 5*\*,000} = *\frac*{400*\*,000}{50*\*,000} = 8 *\text*{ месяцев}*T*=55000−5000400000=50000400000=8 месяцев

Таким образом, срок окупаемости составляет около восьми месяцев, что соответствует поставленной цели — окупить инвестиции в пределах десяти месяцев.

Если необходимо рассчитать допустимый размер инвестиций для достижения срока окупаемости ровно в десять месяцев, то используем преобразованную формулу:

*I*=*T*×(*Sm*−*Cm*) *I* = *T* *\times* (*S\_m* - *C\_m*) *I*=*T*×(*Sm*​−*Cm*​)

Подставляя значения, имеем:

*I*=10×(55 000−5 000)=10×50 000=500 000 рублей*I* = 10 *\times* (55*\*,000 - 5*\*,000) = 10 *\times* 50*\*,000 = 500*\*,000 *\text*{ рублей}*I*=10×(55000−5000)=10×50000=500000 рублей

Это значит, что при текущей экономии и расходах вложения в систему могут составлять до 500 000 рублей, чтобы срок окупаемости не превышал десять месяцев.

Аналогично можно определить минимально необходимую ежемесячную экономию при заданных инвестициях и сроке окупаемости:

*Sm*=*IT*+*CmS\_m* = *\frac*{*I*}{*T*} + *C\_mSm*​=*TI*​+*Cm*​

Для наших параметров:

*Sm*=400 00010+5 000=40 000+5 000=45 000 рублей в месяц*S\_m* = *\frac*{400 *\*,000}{10} + 5*\*,000 = 40*\*,000 + 5*\*,000 = 45*\*,000 *\text*{ рублей в месяц}*Sm*​=10400000​+5000=40000+5000=45000 рублей в месяц

Это означает, что при инвестициях в 400 000 рублей и расходах на поддержку 5 000 рублей ежемесячная экономия должна быть не менее 45 000 рублей для окупаемости за десять месяцев.

Данные расчёты демонстрируют экономическую целесообразность разработки и внедрения виртуального помощника в сервисном центре, обеспечивая снижение затрат и повышение эффективности бизнес-процессов.

3.10 Перспективы развития системы

Схема дополнительных возможностей для ООО «Термо-Мастер сервис» представлена на рисунке 18.

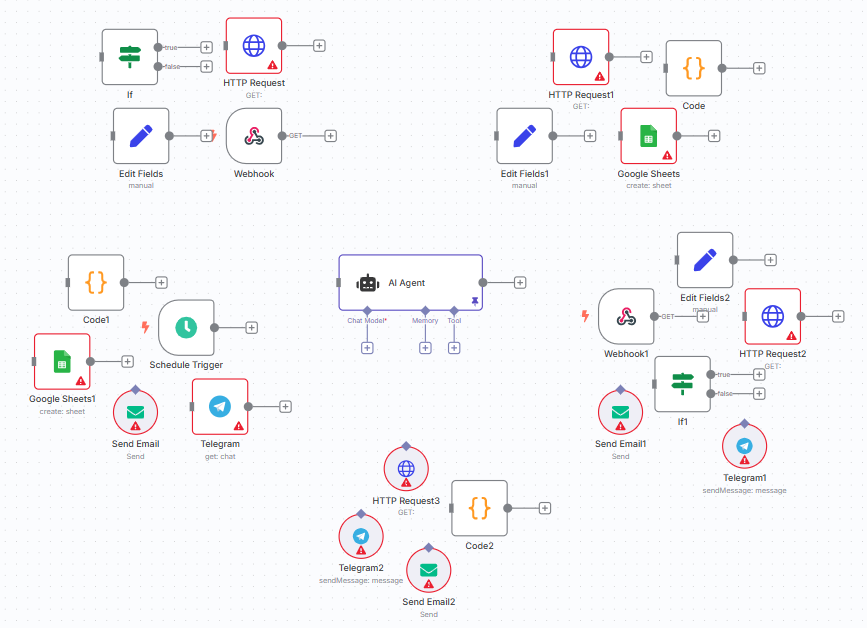


Рисунок 18 – Схема дополнительных возможностей для ООО «Термо-Мастер сервис»

Перспективы развития системы виртуального помощника охватывают широкий спектр направлений, способных существенно повысить её функциональность и сделать взаимодействие с клиентами ещё более эффективным и удобным. Одним из ключевых направлений является внедрение голосового управления и технологии распознавания речи, что позволит пользователям вести диалог с системой естественным языком без необходимости использовать клавиатуру или сенсорный экран. Такой подход значительно упростит процесс получения информации и оформления заказов, особенно для занятых или маломобильных клиентов.

Дальнейшее развитие включает автоматическую генерацию различных отчётов и аналитических сводок. Это обеспечит менеджерам сервисного центра возможность в реальном времени отслеживать статус заказов, анализировать эффективность работы и выявлять узкие места в процессах. Такая прозрачность способствует оперативному принятию управленческих решений и улучшению качества предоставляемых услуг.

Интеграция с геолокационными сервисами позволит отслеживать местонахождение выездных сотрудников, оптимизировать маршруты и сократить время реагирования на заявки. Это особенно важно для сервисных центров с выездным обслуживанием, где своевременность играет ключевую роль в удовлетворённости клиентов. Кроме того, благодаря геолокации можно автоматически информировать клиентов о времени прибытия мастера, что повысит уровень доверия и удобства.

Важной задачей является углубленная интеграция виртуального помощника с корпоративными *CRM*-системами и телефонными платформами. Это позволит обеспечить сквозное управление всей клиентской базой, централизовать данные о заказах и коммуникациях, а также автоматизировать маршрутизацию звонков и сообщений. Благодаря такому подходу повысится качество обслуживания, снизится количество ошибок и упростится контроль за выполнением обязательств.

Автоматизация документооборота является ещё одним перспективным направлением. Внедрение функций автогенерации договоров, счетов и других юридически значимых документов в формате *PDF* позволит сократить время оформления заказов и снизить риск ошибок, связанных с ручным вводом данных. Это значительно разгрузит сотрудников и повысит общую производительность.

Помимо этого, возможна реализация системы самообучения на базе искусственного интеллекта, которая со временем будет анализировать взаимодействия с клиентами, выявлять типичные сценарии и предлагать оптимальные варианты ответов и решений. Такая адаптивность позволит постоянно улучшать качество работы виртуального помощника и делать его более полезным для пользователей.

Также в планах развитие мультиканальной поддержки, включающей не только *Telegram* и *WhatsApp*, но и другие популярные мессенджеры, социальные сети и веб-чат на сайте компании. Это обеспечит клиентам удобный доступ к сервису через любые привычные им каналы связи [3].

Наконец, важным направлением является расширение функционала по автоматизации внутренних процессов, таких как планирование работы сотрудников, управление запасами запчастей и контроль за выполнением гарантийных обязательств. Внедрение подобных возможностей сделает систему универсальным инструментом для комплексного управления сервисным центром.

Все перечисленные направления развития создадут условия для глубокой цифровой трансформации компании, позволят повысить конкурентоспособность, улучшить качество обслуживания клиентов и оптимизировать внутренние бизнес-процессы, что в конечном итоге приведёт к росту лояльности клиентов и увеличению прибыли.

На этапе разработки была создана программная архитектура виртуального помощника, реализована его серверная часть с интеграцией искусственного интеллекта, настроена автоматизация обработки клиентских обращений через мессенджеры. Были определены входные и выходные данные системы, разработаны алгоритмы взаимодействия и логика обработки заказов. Проведено тестирование основных функций для проверки корректности работы и оценки эффективности решения. В результате создано рабочее программное обеспечение, готовое к внедрению и дальнейшему развитию [8].

ЗАКЛЮЧЕНИЕ

В ходе выполнения работы была успешно достигнута поставленная цель - разработан программный виртуальный помощник, предназначенный для автоматизации обработки клиентских обращений и заказов в сервисном центре. Созданное решение позволяет значительно повысить скорость и качество обслуживания клиентов, оптимизировать трудозатраты сотрудников и улучшить общую эффективность работы сервисного центра. Благодаря внедрению виртуального помощника процессы взаимодействия с клиентами становятся более оперативными и структурированными, что способствует росту удовлетворённости пользователей и снижению вероятности ошибок при обработке заказов.

Все основные задачи, поставленные в начале работы, были полностью выполнены. Проведен детальный анализ текущих бизнес-процессов, связанный с обработкой заказов в сервисном центре, что позволило выявить ключевые направления для автоматизации и определить требования к разрабатываемой системе. На основании полученных данных была спроектирована архитектура виртуального помощника, включающая серверную часть с интеграцией искусственного интеллекта, а также построена логика взаимодействия с клиентами и обработки информации.

Разработанный программный виртуальный помощник воплотил в себе основные функции, необходимые для автоматизации процессов — от приёма и классификации обращений до генерации документов и предоставления клиентам необходимых данных. В ходе тестирования были подтверждены стабильность работы системы и её способность эффективно справляться с поставленными задачами.

В итоге, выполненная работа не только подтвердила техническую реализуемость решения, но и продемонстрировала его экономическую эффективность и перспективы дальнейшего развития. Подведены итоги, в которых отражены достигнутые результаты и выявленные возможности для последующего совершенствования системы, что открывает широкие перспективы для цифровой трансформации сервисного центра и повышения его конкурентоспособности на рынке.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. **Документация Telegram Bot API**  
   URL: https://core.telegram.org/bots/api (дата обращения: 01.06.2025);
2. **Telegram Bot API 9.0: Крупнейшее обновление 2025 года**  
   URL: https://habr.com/ru/articles/900126/ (дата обращения: 01.06.2025);
3. **Документация Telegram Gateway API**  
   URL: https://core.telegram.org/gateway/api (дата обращения: 01.06.2025);
4. **Официальная документация n8n**  
   URL: https://docs.n8n.io/ (дата обращения: 01.06.2025);
5. **n8n: Гибкая автоматизация рабочих процессов**  
   URL: https://n8n.io/ (дата обращения: 01.06.2025);
6. **Лучшие варианты хостинга для n8n в 2025 году**  
   URL: https://duforum.in/t/best-n8n-hosting-options-in-2025-the-ultimate-comparison-guide/95898 (дата обращения: 01.06.2025);
7. **Сравнение Zapier и n8n для бизнеса в 2025 году**  
   URL: https://wbtech.ru/lowcode/zapier-or-n8n-for-business-in-2025/ (дата обращения: 01.06.2025);
8. **Полное руководство по разработке API в 2025 году**  
   URL: https://devico.io/blog/complete-guide-to-api-development-in-2025 (дата обращения: 01.06.2025);
9. **API Spring Refining and Equipment Standards Meeting 2025**  
   URL: https://events.api.org/2025-spring-refining-and-equipment-standards-meeting/ (дата обращения: 01.06.2025);
10. **API Fall Refining and Equipment Standards Meeting 2025**  
    URL: https://events.api.org/2025-fall-refining-and-equipment-standards-meeting/ (дата обращения: 01.06.2025);
11. Мартин Фаулер. «Архитектура корпоративных программных приложений» – М.: Вильямс, 2019. – 544 с.
12. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. «Приемы объектно-ориентированного проектирования. Паттерны проектирования» – СПб.: Питер, 2020. – 496 с.
13. Алистер Коберн. «Современные методы описания функциональных требований к системам» – М.: Лори, 2018. – 288 с.
14. Карл Вигерс, Джой Битти. «Разработка требований к программному обеспечению» – М.: Русская Редакция, 2021. – 736 с.
15. Стив Макконнелл. «Совершенный код» – СПб.: БХВ-Петербург, 2022. – 896 с.
16. Ивар Якобсон, Грейди Буч, Джеймс Рамбо. «Унифицированный процесс разработки программного обеспечения» – М.: ДМК Пресс, 2017. – 496 с.
17. Дэвид Вест. «Объектно-ориентированное мышление» – СПб.: Питер, 2020. – 304 с.
18. Крэг Ларман. «Применение UML и шаблонов проектирования» – М.: Вильямс, 2019. – 736 с.
19. Эндрю Таненбаум, Герберт Бос. «Современные операционные системы» – СПб.: Питер, 2021. – 1120 с.
20. Томас Эрл. «Моделирование бизнес-процессов с BPMN 2.0» – М.: Альпина Паблишер, 2020. – 352 с.

Бакалаврская работа выполнена мною самостоятельно. Используемые в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них. Библиографический список содержит 20 наименований.

Один печатный экземпляр и один электронный экземпляр сданы на кафедру ИЭС.

*\_\_\_\_\_\_\_\_\_\_\_\_\_*

(дата)

*\_\_\_\_\_\_\_\_\_\_\_\_\_* *\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

(подпись) (Ф.И.О.)