Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК6)

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОМУ ПРОЕКТУ

НА ТЕМУ:

«Автоматизация работы с отчётами об ошибках в корпоративных банковских системах на основе GPT-технологий»

Студент РК6-32М _____ **Гунько Н.М.**
(Подпись, дата) И.О. Фамилия

Руководитель курсового проекта _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

2024 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой РК6
А.П. Карпенко

«_____» _____ 2024 г.

З А Д А Н И Е
на выполнение научно-исследовательской работы

по теме: Автоматизация работы с отчётами об ошибках в корпоративных банковских системах на основе GPT-технологий

Студент группы РК6-32М

Гунько Никита Макарович
(Фамилия, имя, отчество)

Направленность КП (учебная, исследовательская, практическая, производственная, др.) учебная
Источник тематики (кафедра, предприятие, НИР) предприятие

График выполнения проекта: 25% к 5 нед., 50% к 11 нед., 75% к 14 нед., 100% к 16 нед.

Техническое задание: разработать сетевую программу-прослойку, получающую данные в структурированном виде (JSON, XML) из CRM-системы и отправляющую обработанные данные в API GPT-системы. Обработать в разработанной программе ответ от GPT-системы и передать данные в задачу.

Оформление научно-исследовательской работы:
Расчетно-пояснительная записка на 34 листах формата А4.
Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

Дата выдачи задания «20» сентября 2024 г.

Руководитель курсового проекта _____ **Витюков Ф.А.**
(Подпись, дата) И.О. Фамилия

Студент _____ **Гунько Н.М.**
(Подпись, дата) И.О. Фамилия

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

АННОТАЦИЯ

Расчетно-пояснительная записка содержит 34 с., 3 рис., 6 источников.

Данная работа описывает разработанную программу-прослойку для автоматизированной обработки обращений в банковских системах, реализующегося в рамках исследовательской работы кафедры РК6 МГТУ им. Н. Э. Баумана.

Приведена актуальность работы, архитектура решения, описаны используемые технологии и ключевые этапы обработки. Разработанный прототип снижает нагрузку на сотрудников, ускоряет обработку заявок и стандартизирует их описание.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Современные корпоративные банковские системы ежедневно обрабатывают огромное количество данных, связанных с их функционированием, включая отчеты об ошибках (баг-репорты). Эти данные являются важным источником информации для повышения стабильности и безопасности систем, однако их анализ часто требует значительных временных и человеческих ресурсов. Традиционные методы обработки баг-репортов включают ручной анализ, классификацию и формирование рекомендаций, что является трудоемким и подверженным ошибкам процессом. Это делает актуальным применение современных технологий, таких как языковые модели, для автоматизации данных процессов.

Языковые модели нового поколения, включая GPT-4, Claude и другие, демонстрируют высокий уровень точности и способности к обработке текстовых данных, а также расширяют возможности анализа визуальной информации. Эти модели способны не только интерпретировать и классифицировать текстовые описания, но и генерировать рекомендации, оптимизировать процессы обработки данных и обеспечивать поддержку принятия решений.

В рамках данной работы предлагается исследовать возможности применения языковых моделей для автоматизации обработки отчетов об ошибках в корпоративных банковских системах. Основное внимание уделяется изучению функциональности моделей для анализа текстовых данных и изображений, оценке их интеграции с существующими системами баг-репортов и разработке прототипа взаимодействия с системой на основе API.

Автоматизация обработки баг-репортов предполагает несколько важных этапов: от исследования существующих решений и выбора оптимальных языковых моделей до анализа возможностей API системы баг-репортов Intradesc. Кроме того, необходимо сравнить затраты на использование

облачных и локальных решений, чтобы выбрать подход, соответствующий экономическим и техническим требованиям корпоративных систем.

Цель данной работы – исследовать современные языковые модели, такие как GPT-4 и Claude, для анализа баг-репортов, включая обработку текста и изображений, провести обзор их возможностей, ограничений и стоимости использования с сравнением облачных и локальных решений, а также изучить основные возможности API системы баг-репортов Intradesc, включая поддержку экспорта данных.. Для достижения поставленной цели исследования были выделены следующие задачи:

– Провести анализ современных языковых моделей GPT-4 и Claude, с акцентом на их возможности обработки текста и изображений в контексте баг-репортов.

– Оценить ограничения моделей и их стоимость использования, сравнивая облачные сервисы с локальными решениями.

– Изучить функциональность API системы баг-репортов Intradesc, включая экспорт данных в форматы JSON и XML, и возможности взаимодействия с внешними языковыми моделями.

– Разработать сетевую программу-прослойку, получающую данные в структурированном виде (JSON, XML) из CRM-системы и отправляющую обработанные данные в API GPT-системы.

– Обработать в разработанной программе ответ от GPT-системы и передать данные в CRM-систему.

1. Принципы и подходы к анализу баг-репортов

Баг-репорт – это документ, фиксирующий описание обнаруженной ошибки, её контекст и необходимые данные для воспроизведения и исправления. Чёткая структура баг-репорта повышает эффективность его анализа и позволяет ускорить процесс устранения ошибок. Основными элементами баг-репорта являются:

- **Заголовок**. Этот элемент должен быть кратким и описывать суть проблемы. Например, вместо общего заголовка «Ошибка на странице» лучше использовать более точный «[Авторизация] Ошибка при вводе неверного пароля». Лаконичный и информативный заголовок позволяет быстро понять контекст ошибки без детального прочтения отчёта.
- **Шаги воспроизведения**. Они должны быть чёткими, последовательными и полными, чтобы любой член команды мог воспроизвести проблему. Например:

1. Зайти на страницу авторизации.
2. Ввести неверные учётные данные.
3. Нажать кнопку «Войти». Результатом должно быть точное описание действий, которые неизменно приводят к ошибке.

- **Ожидаемое поведение**. В этом разделе фиксируется, как должна работать система согласно спецификациям. Например: «При вводе неверного пароля должно отобразиться сообщение об ошибке “Неверные учётные данные”».
- **Фактическое поведение**. Здесь описывается, что действительно произошло при выполнении шагов воспроизведения. Например: «После ввода неверного пароля приложение показывает пустую страницу».
- **Дополнительные материалы**. Сюда входят логи, скриншоты, видео или ссылки на записи взаимодействия. Они помогают увидеть детали, которые не всегда можно описать текстом. Например, лог может содержать

исключение: NullReferenceException at line 52, что сразу указывает на источник проблемы.

- **Контекст выполнения**. Включает параметры окружения, такие как версия приложения, операционная система, браузер, а также устройство. Например: «Баг воспроизведён в Chrome 110.0.5481.100 на Windows 10 x64». Это особенно важно, когда проблема может быть связана с определённой платформой.
- **Приоритет и статус**. Приоритет определяет, насколько критична ошибка для системы: блокирующая, высокая, средняя или низкая. Статус отражает текущее состояние задачи: новый, в работе, исправлен, закрыт.

Каждый из этих элементов важен для всестороннего анализа. Если хотя бы одна часть отсутствует или выполнена некачественно, это может замедлить процесс поиска и устранения проблемы [1].

1.1. Проблемы традиционного подхода

Традиционные методы обработки баг-репортов предполагают ручной анализ, что связано с рядом ограничений. Одной из главных проблем является **человеческий фактор**. Эксперты, анализирующие репорты, могут ошибаться, недооценивать значимость ошибок или терять важные детали. Например, при нехватке времени дубликаты репортов могут не быть выявлены, что приводит к увеличению объёма задач.

Сложность масштабирования также является серьёзной проблемой. В условиях крупных проектов и сложных систем количество багов может достигать тысяч, и их ручная обработка становится неподъёмной. Переход к автоматизированным инструментам позволяет снять часть этой нагрузки, но далеко не все традиционные системы баг-трекинга, такие как Jira или Bugzilla, поддерживают такие технологии.

Ещё одним слабым местом является **выявление дубликатов баг-репортов**. Разные пользователи и тестировщики могут сообщать об одной и той же проблеме, формулируя её по-разному. Например, один баг может быть

описан как «Ошибка при сохранении формы», а другой как «Форма не сохраняет данные». Такие случаи требуют продвинутого анализа текста, который ручным способом выполнить трудно.

Наконец, недостаток **интеграции с передовыми технологиями** ограничивает возможности современных инструментов. Многие системы не поддерживают анализ текстов с использованием машинного обучения, обработку мультимодальных данных (текст + изображения), а также автоматическую классификацию и оценку багов.

1.2. Автоматизация анализа

Использование крупных языковых моделей, таких как GPT или подобных, для автоматизации обработки баг-репортов предлагает значительные преимущества, устраняя ограничения традиционного подхода. Вот ключевые направления применения:

1. **Классификация баг-репортов**. Модели могут автоматически определять типы ошибок (например, баг интерфейса, проблемы с логикой или производительностью), а также классифицировать их по степени приоритетности. Например, описания вроде «Форма отправки данных не работает» модель классифицирует как проблему бизнес-логики с высоким приоритетом.

2. **Выявление дубликатов**. Алгоритмы анализа текста способны сравнивать описания и находить семантические совпадения, даже если формулировки различаются. Например, репорты «Кнопка “Сохранить” не работает» и «Не сохраняются изменения» будут идентифицированы как дубликаты.

3. **Генерация рекомендаций**. На основе анализа предыдущих решений языковые модели могут предлагать разработчикам возможные пути устранения ошибок. Например, если определённый баг связан с исключением NullPointerException, модель может предложить проверить, инициализированы ли все переменные.

4. **Обработка мультимодальных данных**. Современные языковые модели могут работать не только с текстом, но и с изображениями или видео. Например, анализируя скриншоты, модель может извлечь текст ошибки или сопоставить изображение с известными проблемами в базе данных.

5. **Автоматическое заполнение информации**. Лингвистические модели способны дополнять неполные баг-репорты, добавляя стандартные формулировки или запрашивая недостающую информацию. Например, если в отчёте отсутствует раздел с ожидаемым поведением, модель может автоматически сформулировать его, опираясь на спецификацию.

Пример работы языковой модели

Предположим, поступает баг-репорт с заголовком «Ошибка при загрузке профиля». Языковая модель анализирует текст, обнаруживает, что в шаге воспроизведения упоминается поле с изображением, и классифицирует проблему как баг, связанный с бизнес-логикой. Затем она находит схожие ошибки в системе и добавляет их в репорт как возможные дубликаты. В результате модель предлагает решение: «Проверьте путь к изображению и настройку разрешений».

Автоматизация с использованием языковых моделей решает ключевые проблемы традиционного подхода. Она повышает точность, устраняет дублирование усилий и ускоряет процесс обработки больших объёмов данных. Это делает языковые модели незаменимым инструментом для команд, работающих над сложными проектами [2].

2. Анализ возможностей современных языковых моделей

Современные языковые модели, такие как GPT-4 и Claude, представляют собой мощные инструменты, разработанные для обработки сложных текстовых данных, мультимодальных входов (текст и изображения) и выполнения широкого спектра задач, включая анализ, генерацию текста, классификацию и интерпретацию. Эти модели основываются на архитектуре Transformer и применяют механизм внимания, что позволяет учитывать

контекст на протяжении всей обработки данных. GPT-4 и Claude разрабатывались с разными акцентами: первая ориентирована на универсальность и точность, а вторая – на безопасность и соблюдение этических принципов.

GPT-4 демонстрирует высокую производительность в задачах анализа больших текстовых массивов, генерации контента и обработке мультимодальных данных. Модель способна интерпретировать текстовые баг-репорты, выявлять закономерности и генерировать предложения по устранению проблем. Claude, в свою очередь, специализируется на безопасной и точной обработке данных в корпоративных системах, что делает её особенно полезной для анализа информации, связанной с бизнес-процессами, где важно учитывать соблюдение нормативных требований.

2.1. Возможности GPT-4

GPT-4 является одной из наиболее мощных языковых моделей, доступных сегодня. Её ключевая особенность – возможность работать с большими объёмами текста, адаптируясь к различным сценариям. Модель не только успешно классифицирует данные, но и способна анализировать мультимодальные входы, такие как текст в сочетании с изображениями, что особенно важно в задачах анализа баг-репортов.

Модель GPT-4 отличается высокой масштабируемостью и способностью обрабатывать большие объемы текстовых данных. Она обучена на обширных текстовых корпусах, включающих различные типы текстов, от научных статей и новостных материалов до художественных произведений и социальных сетей. Это разнообразие позволяет модели понимать и генерировать текст на различных уровнях сложности и стиля, что делает её универсальным инструментом для обработки естественного языка [3, 4].

2.1.1. Преимущества

GPT-4 отличается универсальностью и высокой точностью. Она эффективно классифицирует текстовые баг-репорты, разделяя их по

категориям, таким как ошибки интерфейса, логические сбои или проблемы производительности. Например, в случае баг-репорта с описанием «Ошибка 404 при загрузке страницы профиля» модель может предложить проверить корректность маршрутов API или параметры конфигурации сервера.

Ещё одна сильная сторона GPT-4 – её способность работать с мультимодальными данными. Модель может анализировать текстовые описания ошибок в сочетании со скриншотами, на которых отображаются тексты ошибок или неудачно отрисованные элементы интерфейса. Например, при анализе баг-репорта, включающего скриншот с текстом «Ошибка подключения: TimeOut», GPT-4 успешно извлекает текст с изображения, сопоставляет его с описанием и генерирует рекомендации по устранению ошибки.

Кроме того, GPT-4 может выявлять скрытые закономерности в данных и находить дубликаты баг-репортов. Это особенно полезно в ситуациях, когда система баг-трекинга переполнена схожими ошибками, описанными разными пользователями.

2.1.2. Ограничения

Несмотря на мощь модели, GPT-4 требует значительных вычислительных ресурсов для работы. Это может стать проблемой при обработке большого объёма баг-репортов в реальном времени. Ещё одним ограничением является зависимость от качества обучающих данных. Если модель обучена на ошибочных или неполных данных, это может негативно сказаться на её производительности.

2.2. Возможности Claude

Claude разработан с акцентом на обеспечение безопасности и точности обработки данных, что делает его незаменимым инструментом для корпоративных задач. Модель особенно полезна в контексте анализа баг-репортов, связанных с бизнес-процессами, поскольку она способна учитывать

тонкости нормативных требований и сложные взаимосвязи в корпоративных системах.

2.2.1. Преимущества

Claude демонстрирует отличные результаты в задачах интерпретации текста и анализа бизнес-логики багов. Например, при анализе баг-репорта с описанием «Сбой в процессе аутентификации пользователя» модель может предложить проверить корректность интеграции с системами управления доступом и назначить ответственную команду для устранения ошибки.

Одной из ключевых возможностей Claude является генерация решений с учётом корпоративных требований. Например, если баг-репорт связан с платёжной системой, модель может предложить проверить соответствие операции правилам AML (Anti-Money Laundering) и другим регуляторным стандартам. Это делает Claude особенно полезным для компаний, работающих в финансовом секторе.

Claude также эффективно выявляет дубликаты багов, анализируя текстовые описания и связанные метаданные. Модель способна объединить схожие баг-репорты, что помогает сократить объём работы аналитиков [5].

2.2.2. Ограничения

Одним из главных ограничений Claude является его фокус на безопасность и этичность, что иногда может снижать производительность в задачах, требующих гибкости. Например, в ситуациях, где требуется интерпретация неоднозначных данных, модель может проявлять излишнюю консервативность. Кроме того, как и GPT-4, Claude требует значительных вычислительных ресурсов.

2.3. Возможности обработки данных при анализе баг-репортов

Современные языковые модели GPT-4 и Claude обладают широкими возможностями обработки как текстовых, так и мультимодальных данных (включая изображения и текстовые файлы). Эти функции позволяют

эффективно анализировать баг-репорты, объединяя текстовую информацию, визуальные материалы (скриншоты), журналы ошибок и другие контекстные данные. Такой комплексный подход позволяет автоматизировать процесс классификации ошибок, приоритизации и назначения ответственных лиц или команд, а также предоставляет рекомендации по устранению проблем.

2.3.1. Текстовые данные

Текстовые данные, как основной компонент баг-репорта, содержат критически важную информацию, включая:

- **Описание ошибки**: объяснение природы проблемы с деталями.
- **Шаги воспроизведения**: последовательность действий, приводящая к ошибке.
- **Ожидаемое поведение**: информация о том, как система должна работать.
- **Фактическое поведение**: описание текущего состояния, включая сбои или неверные результаты.
- **Контекст бизнес-процесса**: информация о функциональности, в которой возникла ошибка (например, авторизация, платёжная операция или отчётность).

Языковые модели способны структурировать текст, выделять ключевые аспекты и предлагать возможные решения. Например, при анализе сложного описания модели могут уточнить, что ошибка связана с логикой расчёта скидки в корзине покупок.

2.3.2. Изображения

Баг-репорты часто содержат скриншоты, которые иллюстрируют проблему. Современные модели, такие как GPT-4 с поддержкой мультимодальных данных, позволяют:

1. **Извлекать текст с изображения** (например, коды ошибок, названия полей или сообщений об ошибках).

2. **Анализировать визуальные элементы интерфейса** (например, расположение кнопок, шрифты, некорректное отображение элементов).

3. **Соотносить визуальные данные с текстом баг-репорта** для уточнения проблемы и предоставления более точных рекомендаций.

Например, если баг-репорт содержит скриншот с ошибкой 404, модель может предложить проверить корректность URL или конфигурацию маршрутизации на сервере.

2.3.3. Текстовые файлы (логи)

Текстовые файлы с логами – это источник детализированной технической информации. Языковые модели могут:

- **Выявлять коды ошибок** и другие значимые метрики.
- **Искать закономерности** в логах, такие как повторяющиеся тайм-ауты или исключения.
- **Сопоставлять записи логов с текстовыми данными** баг-репорта, чтобы уточнить источник проблемы.

Например, если в журнале зафиксировано превышение времени ожидания, модель может порекомендовать пересмотреть параметры тайм-аутов или проанализировать производительность серверных запросов.

2.4. Формирование запроса к языковой модели

Запрос к языковой модели или “промпт” – это специально сформулированный запрос, направляемый языковой модели, с целью получения точного и структурированного ответа. Он включает инструкции для анализа и обработки входных данных, а также указания на формат вывода, необходимый для дальнейшей автоматической обработки. Грамотно составленный “промпт” позволяет оптимизировать взаимодействие с языковыми моделями, обеспечивая высокую точность и релевантность результатов.

Создание эффективного “промпта” для языковой модели является ключевым этапом для автоматизации обработки баг-репортов. Запрос должен

включать все необходимые инструкции для анализа текста, изображений и логов, а также указания на формат структурированного вывода. Такой подход позволяет интегрировать результат анализа в вашу CRM-систему без дополнительных преобразований данных.

Запрос должен содержать ясные инструкции, описывающие шаги анализа. Это необходимо для того, чтобы модель могла:

- Классифицировать баг-репорт по типу ошибки.
- Определять уровень приоритета и серьёзность проблемы.
- Анализировать бизнес-процесс, на который влияет ошибка.
- Идентифицировать исполнителя или команду, ответственную за решение проблемы.
- Обрабатывать мультимодальные данные (например, текст, изображения, логи).

Для автоматизации процесса необходимо заранее задать формат ответа, например, в виде JSON. Это позволяет избежать неоднозначностей и упрощает последующую обработку данных. Структура вывода должна включать ключевые параметры:

- Тип ошибки.
- Приоритет.
- Затронутый бизнес-процесс.
- Ответственная команда.
- Рекомендации по устранению.
- Результаты анализа приложенных данных (если есть).

Так как баг-репорты могут содержать разные типы данных (текст, изображения, логи), “промпт” должен быть адаптирован для обработки всех этих форматов. Важно предусмотреть обработку опциональных данных: если изображений или логов нет, анализ проводится только по тексту.

Пример запроса для анализа и классификации баг-репорта:

```
Вы выступаете в роли аналитика баг-репортов. Ваша задача — классифицировать
ошибку, определить её приоритет, проанализировать бизнес-процесс, затронутый
```

````
проблемой, и предложить рекомендации для её устранения. Результат должен быть представлен в формате JSON для автоматической обработки.

**Входные данные:**

1. **Описание ошибки**:
   [Текст баг-репорта, включая шаги воспроизведения, ожидаемое и фактическое поведение].

2. **Скриншоты** (опционально):
   [Приложенные изображения, если есть].

3. **Логи** (опционально):
   [Фрагменты журналов ошибок, если доступны].

**Инструкции по анализу:**
1. Определите тип ошибки: логическая, интерфейсная, связанная с производительностью, интеграционная или другая.
2. Укажите приоритет ошибки: низкий, средний, высокий, блокирующий.
3. Определите затронутый бизнес-процесс (например, авторизация, обработка платежей, отчётность, пользовательский интерфейс).
4. Назначьте команду или роль, ответственную за устранение ошибки (например, backend-разработчики, frontend-разработчики, DevOps-инженеры).
5. Если приложены скриншоты, выполните анализ изображений (например, визуальные дефекты, сообщения об ошибке).
6. Если приложены логи, выделите ключевую информацию (например, коды ошибок, текстовые сообщения).
7. Сформируйте рекомендации по устранению ошибки с указанием конкретных действий.

**Формат вывода:**

```json
{
    "type": "<тип ошибки>",
    "priority": "<приоритет>",
    "business_process": "<затронутый бизнес-процесс>",
    "responsible_team": "<ответственная команда>",
    "analysis": {
        "description": "<анализ текстового описания>",
        "screenshot_analysis": "<анализ изображения (если есть)>",
        "log_analysis": "<анализ логов (если есть)>"
    },
    "recommendations": [
        "<рекомендация 1>",
        "<рекомендация 2>"
    ]
}
````

Пример ввода:
````

```
Описание ошибки: "Пользователь не может авторизоваться. После ввода логина и
пароля появляется сообщение 'Ошибка 500'."
Скриншоты: [Приложено изображение, где видно сообщение 'Ошибка сервера.']
Логи: 2023-01-24 12:45:32 [ERROR] Connection timeout: auth_service
```

Пример вывода:

```
{
    "type": "логическая",
    "priority": "блокирующий",
    "business_process": "авторизация",
    "responsible_team": "backend-разработчики",
    "analysis": {
        "description": "Пользователь не может авторизоваться. Сообщение 'Ошибка
500' свидетельствует о проблемах с сервером авторизации.",
        "screenshot_analysis": "На изображении видно сообщение 'Ошибка
сервера'.",
        "log_analysis": "Лог указывает на тайм-аут подключения к серверу
авторизации."
    },
    "recommendations": [
        "Проверить доступность сервера авторизации.",
        "Перепроверить параметры тайм-аутов в конфигурации сервера."
    ]
}
```

Полученный JSON-ответ можно использовать для создания задачи в CRM. Например:

1. Извлечь данные из JSON:
    - Тип ошибки.
    - Приоритет.
    - Ответственную команду.
    - Рекомендации.

2. Вызвать API вашей CRM для создания задачи, передав извлечённые данные.

3. При необходимости прикрепить скриншоты и логи к созданной задаче.

Такой промпт и формат ответа обеспечивают чёткую и структурированную обработку баг-репортов, упрощая их анализ и последующее управление задачами.

3. Интеграция с системой Intradesc

Intradesc – это система учета заявок с веб-интерфейсом. Система подходит для обработки заявок клиентов, постановки задач, организации полноценной службы Service Desk или оказания аутсорсинговых услуг. Благодаря наличию CRM-функциональности Intradesk можно использовать для автоматизации продаж.

Указанная система предоставляет мощное API для автоматизации процессов обработки заявок, баг-репортов и задач. Это API предназначено для взаимодействия с различными компонентами, включая системы управления задачами, языковые модели и внешние инструменты автоматизации. Ключевые особенности системы включают:

1. **Работа с заявками**: API позволяет получать, изменять, фильтровать и удалять заявки. Каждая заявка содержит уникальный идентификатор, текстовое описание проблемы, прикрепленные файлы и статус обработки.

2. **Поддержка бизнес-процессов**: Все заявки относятся к конкретным бизнес-процессам, поддерживается назначение исполнителей, установка приоритетов, комментариев и сроков выполнения.

3. **Авторизация**: Поддерживаются два основных способа авторизации – через API-ключи и токены. API-ключи подходят для простых задач, но использование токенов более безопасно для сценариев с длительными сессиями.

4. **Фильтрация данных**: Система поддерживает OData-условия для гибкого отбора задач, таких как фильтрация по статусу, группе исполнителей, сервису или приоритету.

5. **Обработка файлов**: API позволяет прикреплять файлы к заявкам и комментировать их. Например, можно прикреплять логи или изображения, иллюстрирующие проблему.

6. **Взаимодействие с исполнителями и заявителями**: Через API можно передавать заявки на исполнение, назначать группы или отдельных

исполнителей, а также инициировать взаимодействие с отправителем заявки для уточнения деталей [6].

3.1. Автоматизация работы с заявками

Языковые модели (например, GPT-4) могут быть интегрированы в систему для автоматизации обработки баг-репортов и задач. Это включает их классификацию, уточнение деталей через автоматизированные вопросы, генерацию предложений по исправлению и назначение исполнителей. Далее представлен алгоритм такой интеграции.

Шаг 1. Получение заявок из системы Intradesc

Система Intradesc предоставляет API для получения списка заявок. Чтобы получить данные, необходимо выполнить GET-запрос к API.

Пример запроса:

```
GET
https://apigw.intradesk.ru/tasklist/odata/tasks?ApiKey=YOUR_API_KEY&$filter=statu
s eq 22055&$orderby=updatedat desc&$top=10
```

Этот запрос возвращает первые 10 заявок в статусе “открыта”, отсортированные по дате обновления. Используйте токен для авторизации или укажите ApiKey в строке запроса.

Шаг 2. Анализ описания задачи с использованием языковой модели

После получения заявки текст ее описания передается языковой модели для анализа. Пример задачи для языковой модели:

1. Определить, к какой категории или бизнес-процессу относится проблема.
2. Выявить недостающие данные или неясности в описании.
3. Сформулировать вопросы для уточнения задачи у заявителя (если необходимо).

Пример сценария обработки на уровне запроса:

```
POST https://api.openai.com/v1/chat/completions
Content-Type: application/json
Authorization: Bearer YOUR_GPT_API_TOKEN

{
```

```
  "model": "gpt-4",
  "messages": [
    {"role": "system", "content": "Вы помогаете анализировать баг-репорты…"},
    {"role": "user", "content": "Описание проблемы…"}
  ],
  "max_tokens": 500
}
```

Ответ модели содержит предложение по классификации заявки, уточняющие вопросы и возможные шаги для решения.

Шаг 3. Автоматическое уточнение деталей

Если языковая модель определила, что описание заявки недостаточно для ее выполнения, она формирует вопросы. Эти вопросы отправляются заявителю с помощью функции добавления комментариев.

Пример запроса на добавление комментария:

```
PUT https://apigw.intradesk.ru/changes/tasks?ApiKey=YOUR_API_KEY
Content-Type: application/json

{
    "number": 12345,
    "blocks": {
        "comment": "{\"value\":\"Можете уточнить, какой именно сервер и когда
последний раз он работал корректно?\"}"
    }
}
```

Шаг 4. Классификация и назначение исполнителя

После анализа заявки модель классифицирует проблему и определяет, к какому бизнес-процессу она относится. Затем заявка передается в соответствующую группу исполнителей или назначается конкретному сотруднику.

Пример запроса на назначение исполнителя:

```
PUT https://apigw.intradesk.ru/changes/tasks?ApiKey=YOUR_API_KEY
Content-Type: application/json

{
    "number": 12345,
    "blocks": {
        "executor": "{\"value\":{\"userid\":583437,\"groupid\":176284}}",
        "status": "{\"value\":22058}"  // "В работе"
    }
```

```
}
```

Шаг 5. Отправка обновленного статуса и результатов анализа

После назначения заявки исполнителю, система может автоматически отправить обновленную информацию заявителю. Это может включать подтверждение получения заявки, указание статуса и контактных данных исполнителя.

Пример запроса:

```
PUT https://apigw.intradesk.ru/changes/tasks?ApiKey=YOUR_API_KEY
Content-Type: application/json

{
    "number": 12345,
    "blocks": {
        "comment": "{\"value\":\"Заявка принята в работу. Ответственный: Иванов
Иван (Первая линия поддержки).\"}",
        "status": "{\"value\":22058}"  // "В работе"
    }
}
```

Шаг 6. Дополнительная обработка (например, прикрепление файлов)

Если языковая модель определила необходимость приложить дополнительную информацию (например, скриншоты, файлы с логами), файлы загружаются в систему. После загрузки файлы прикрепляются к описанию заявки.

Пример загрузки файла:

```
POST
https://apigw.intradesk.ru/files/api/tasks/0/files/target/Description?ApiKey=YOUR
_API_KEY
Content-Type: multipart/form-data

--boundary
Content-Disposition: form-data; name="file"; filename="log.txt"
Content-Type: text/plain

(содержимое файла)
--boundary--
```

Шаг 7. Постановка задач на основе анализа

Если заявка требует дальнейшего уточнения или оценки, система автоматически формирует подзадачи для других исполнителей. Это может быть сделано с помощью функции создания дочерних заявок.

Пример связывания заявок:

```
PUT https://apigw.intradesk.ru/changes/taskrelations?ApiKey=YOUR_API_KEY
Content-Type: application/json

{
    "Tree": {
        "Number": 12345,
        "Children": [
            {"Number": 67890, "Children": []}
        ]
    }
}
```

Этот алгоритм позволяет полностью автоматизировать обработку баг-репортов: от получения данных до назначения исполнителей и уточнения деталей через комментарии. Языковые модели выполняют ключевую роль в анализе и классификации заявок, а система Intradesc обеспечивает гибкие инструменты для управления задачами и интеграции с внешними сервисами.

4. Автоматизирующая программа-прослойка

Автоматизирующая программа-прослойка (АПП) предназначена для упрощения обработки обращений (багов программного обеспечения) от банков и их эффективной маршрутизации внутри компании. Данная система минимизирует ручные действия сотрудников, анализирует поступающие обращения, уточняет недостающие данные, передает их в OpenAI API для генерации структурированных отчетов и направляет обработанные заявки в соответствующие отделы компании.

До внедрения АПП процесс обработки обращений (багов) от банков был полностью ручным (рис. 1).

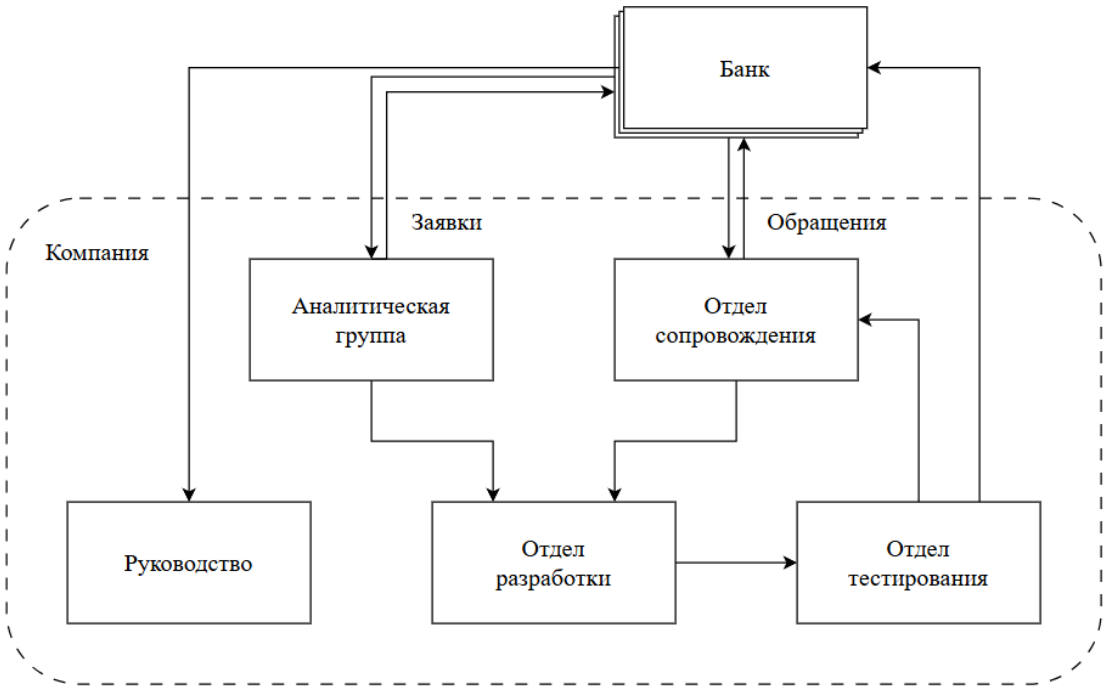

Рисунок 1 – Схема взаимодействия отделов компании до внедрения АПП

Банк отправляет обращение в отдел сопровождения, где сотрудники вручную проверяют описание проблемы, анализируют ее полноту и, если необходимо, запрашивают у банка дополнительную информацию. Затем обращение передается в аналитическую группу, которая оценивает его сложность и определяет, в какой отдел компании направить для решения. Если требуется доработка, обращение поступает в отдел разработки, а после реализации исправлений передается в отдел тестирования.

Основные проблемы процесса до автоматизации:

- Высокая нагрузка на отдел сопровождения – вручную анализировать и уточнять обращения занимает много времени.
- Задержки в обработке – если обращение неполное, его возврат на уточнение и повторная передача создают задержки.
- Ручное распределение – аналитическая группа вручную принимает решение, какой отдел компании займется решением проблемы.
- Отсутствие единого формата – описания ошибок могут быть разными, что затрудняет обработку.

После внедрения АПП в процесс обработки обращений был добавлен автоматический анализатор, который теперь обрабатывает все входящие обращения (рис. 2).

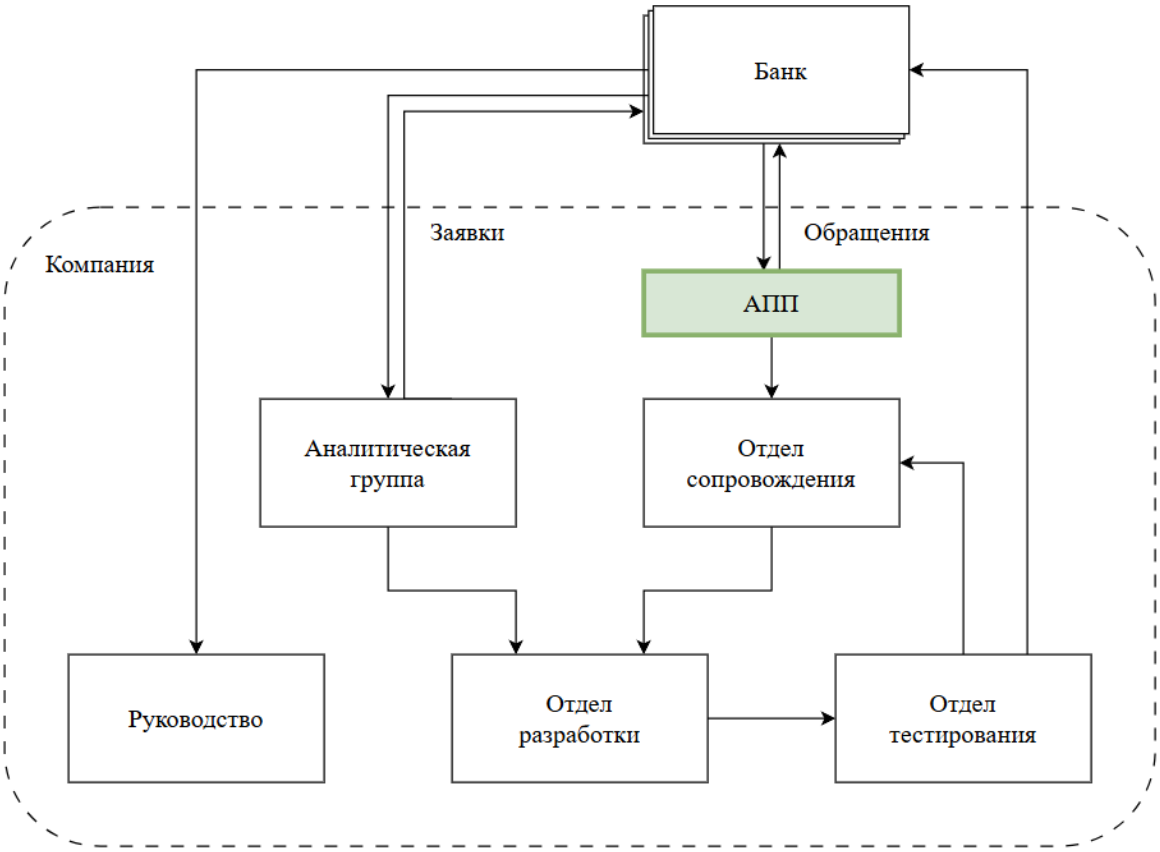

Рисунок 2 – Схема взаимодействия отделов компании после внедрения АПП

Автоматический опрос Intradesc API выполняется с заданной периодичностью, позволяя оперативно проверять наличие новых обращений. Если в поступившем обращении отсутствует важная информация, система автоматически запрашивает ее у банка через Intradesc API. Данный механизм снижает количество неинформативных обращений и исключает необходимость их ручной доработки (рис. 3).

Программа интегрирована с OpenAI API, что позволяет анализировать текст обращения и формировать структурированное описание проблемы. Это способствует стандартизации заявок и снижению времени на их обработку. После выполнения анализа система автоматически определяет, в какой отдел (аналитики, сопровождение, разработка или тестирование) необходимо

направить обращение, исключая необходимость ручного распределения (рис. 3).

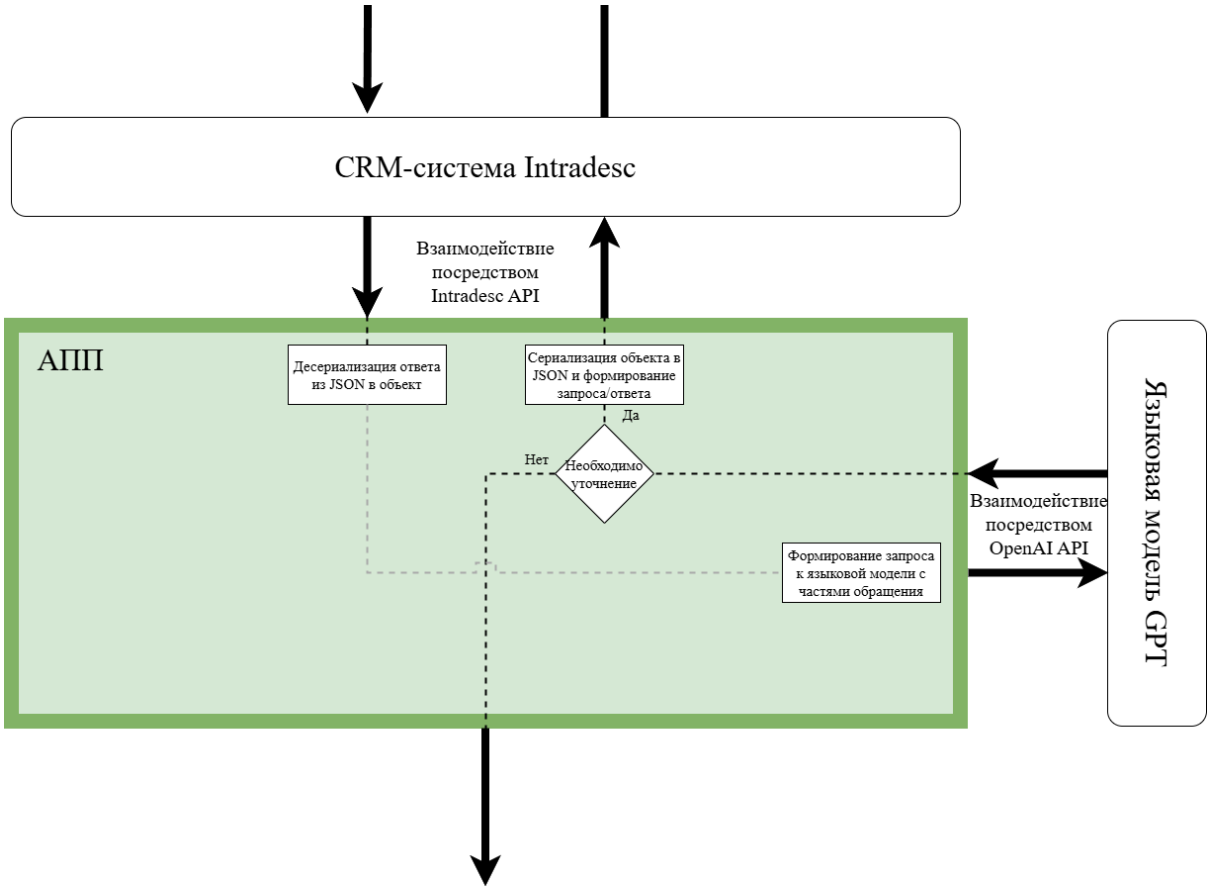

Рисунок 3 – Внутренняя логика АПП

Для обеспечения прозрачности работы ведется логирование всех действий, включая обработку входящих данных, запросы на уточнение информации, анализ с использованием OpenAI API и передачу заявок в соответствующие отделы компании. Это позволяет вести аудит работы системы и оперативно выявлять возможные проблемы в обработке обращений.

Программа реализована с использованием C# и работает на платформе .NET 8.0. Она взаимодействует с Intradesc API для получения обращений и их обновления, а также с OpenAI API для анализа и форматирования данных.

Основные технологии:

- C# (.NET 8.0, Console Application) – основной язык и среда выполнения.
- Intradesc API – для получения и обновления обращений из CRM-системы.

- OpenAI API – для обработки естественного языка и генерации

4.1. Получение данных из CRM-системы Intradesc

Программа периодически отправляет запрос к Intradesc API, проверяя наличие новых обращений. Запрос выполняется в формате HTTP GET с указанием API-ключа и необходимых параметров фильтрации.

Пример запроса:

```
GET
https://apigw.intradesk.ru/tasklist/odata/v3/tasks?ApiKey=API_KEY&$filter=status
eq 22055
```

Где 22055 – статус “Открыта”.

Программа получает JSON-ответ от API, содержащий список обращений с основными параметрами:

- ID обращения
- Название и описание проблемы
- Текущий статус
- Исполнитель
- Дополнительные метаданные (теги, приоритет и т. д.)

Пример ответа в формате JSON:


```
{
    "@odata.count": 10000,
    "value": [
        {
            "id": 3398028,
            "tasknumber": 24484,
            "customerid": 5002,
            "status": 39762,
            "priority": 19947,
            "name": "Оплата штрафов ФЛ",
            "description": "При автоматическом поиске штрафов при заходе в ДБО
находится штраф, но оплатить его не могу...",
            "initiator": 214994,
            "executor": 290982,
            "creator": 214994,
            "createdby": "Фогель Михаил Александрович",
            "createdat": "2025-02-07T10:17:35.0699330Z",
            "updatedat": "2025-02-16T19:41:08.7661321Z",
            "updatedby": "Бойков Алексей Вадимович",
            "service": 44760,
```

```
            "servicename": "Поддержка",
            "tasktype": 13495,
            "resolutiondateplan": "2025-03-07T10:00:00.0000000Z",
            "tags": [],
            "additionalfields": {
                "data": [
                    {
                        "alias": "addfield_versiia_obnovleniiaString",
                        "value": { "stringvalue": "1579p5" }
                    },
                    {
                        "alias": "addfield_tip_obrashcheniiaSingleSelect",
                        "value": { "stringvalue": "ОШИБКА" }
                    }
                ]
            },
            "attachments":
"[{\"Id\":\"67af0781524abb311bbab4e8\",\"Name\":\"UbsRBS_25021413.zip\"},...]",
            "servicefullname": "Поддержка",
            "createdchannel": "web"
        }
    ]
}
```


4.2. Десериализация данных

После получения ответа от Intradesc API, программа десериализует JSON-данные в объект внутреннего формата. Десериализация выполняется с использованием System.Text.Json в C#. На этом этапе данные становятся доступными для анализа и обработки.

4.3. Анализ полноты данных

После десериализации программа анализирует, содержит ли обращение всю необходимую информацию. Проверяется наличие обязательных ответов на вопросы:

Что делал пользователь?

Что он получил?

Что должен был получить?

Почему этого не произошло? (если применимо)

Если каких-либо данных не хватает, система отправляет запрос на уточнение в Intradesc API, изменяя статус обращения или добавляя комментарий с просьбой предоставить недостающие сведения.

Пример запроса на уточнение:

```
PUT https://apigw.intradesk.ru/changes/v3/tasks/?ApiKey=API_KEY
{
  "number": 12345,
  "blocks": {
    "comment": {"value": "Пожалуйста, уточните: что должно было произойти?"}
  },
  "Channel": "api"
}
```

Если данные полные, программа переходит к следующему этапу.

4.4. Обращение к OpenAI API

Когда обращение содержит достаточное количество информации, программа формирует запрос к языковой модели GPT для детального анализа и формулирования структурированного ответа.

Программа формирует текстовый запрос, передавая модель информацию о проблеме. Запрос содержит:

- Описание проблемы
- Исходные данные из обращения
- Инструкцию по формату ответа

Пример запроса к OpenAI API:

```
{
    "model": "gpt-4",
    "prompt": "Проанализируй обращение: 'Ошибка при формировании отчета'.\nЧто
делаем? Что получаем? Что должны получить? Почему этого не происходит?",
    "max_tokens": 150
}
```

В ответ GPT возвращает структурированный анализ ошибки. Пример ответа и он передается программе для дальнейшей обработки.

4.5. Формирование окончательного ответа

На основе полученного результата программа сериализует объект в JSON и формирует запрос к Intradesc API, обновляя обращение и направляя его в соответствующий отдел.

4.6. Логирование всех операций

Программа ведет логирование всех этапов обработки обращений, включая:

- Время получения новых данных
- Отправленные запросы в Intradesc API
- Результаты анализа OpenAI API
- Обновленные статусы обращений

Пример лог-файла:

```
2024-02-18 12:30:00 [INFO] Получено 3 новых обращения.
2024-02-18 12:30:05 [INFO] Отправлен запрос в OpenAI для заявки #31.
2024-02-18 12:30:07 [INFO] Обновлена заявка #31, статус: В работе.
```

Логирование позволяет отслеживать обработку обращений и выявлять возможные сбои.

Автоматизирующая программа-прослойка (АПП) обеспечивает полный цикл обработки обращений от получения данных из Intradesc API до анализа и передачи их в нужный отдел. Применение OpenAI API позволяет формировать более точные и детализированные описания ошибок, а автоматическое уточнение данных снижает нагрузку на сотрудников. Логирование операций обеспечивает контроль и прозрачность процесса. В результате обработки обращений стали **быстрее, точнее и эффективнее**, что позволило снизить объем ручного труда и сократить задержки в анализе ошибок.

ЗАКЛЮЧЕНИЕ

Выполненная работа была направлена на исследование возможностей современных языковых моделей для автоматизации обработки баг-репортов в корпоративных банковских системах. Основное внимание уделялось анализу функциональности таких моделей, как GPT-4 и Claude, а также их интеграции с системой учета заявок Intradesc. Проведенный анализ и практическая часть работы позволили получить разностороннее представление о потенциале языковых моделей в области обработки баг-репортов.

В рамках исследования было показано, что современные языковые модели обладают широкими возможностями для автоматизации сложных процессов обработки ошибок, включая классификацию, выявление дубликатов, генерацию рекомендаций и взаимодействие с пользователями через уточняющие запросы. Модель GPT-4 продемонстрировала высокую точность и универсальность в обработке текстовых данных, включая описание багов и анализ логов, а также способность к мультимодальной обработке данных, что особенно важно при работе с изображениями, иллюстрирующими ошибки. В то же время модель Claude была отмечена за безопасность обработки данных и способность учитывать нормативные и бизнес-требования, что делает её подходящей для задач финансовых организаций.

Анализ API системы Intradesc выявил её широкие возможности для интеграции с языковыми моделями. API предоставляет инструменты для автоматизации работы с заявками, включая их классификацию, назначение исполнителей и обработку вложений. В работе было разработано решение, которое, используя API системы, позволяет реализовать полный цикл автоматизации: от получения баг-репорта до назначения ответственного лица и уточнения деталей через автоматизированные комментарии. Особое внимание уделено алгоритмизации работы, включая детальный разбор форматов запросов к API, взаимодействие с языковой моделью и создание структурированных результатов для последующей обработки.

Для реализации автоматизированного процесса была разработана программа-прослойка, которая выполняет следующие функции:

- Получение данных в структурированном виде (JSON) из CRM-системы Intradesc посредством API.
- Анализ полноты данных и формирование уточняющих запросов при необходимости.
- Взаимодействие с языковой моделью OpenAI API для генерации структурированного описания ошибок и рекомендаций.
- Обратная передача обработанных данных в CRM-систему с автоматическим назначением исполнителей.
- Логирование всех этапов работы для контроля и мониторинга обработки обращений.

Разработанная программа-прослойка позволила значительно снизить нагрузку на сотрудников, ускорить обработку обращений и обеспечить стандартизацию описаний ошибок. Важным преимуществом является автоматическое выявление недостающих данных и генерация уточняющих запросов, что исключает задержки, связанные с ручной обработкой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. How to Write a Good Bug Report: Best Practices and Tips // TestLodge Blog [Электронный ресурс]. Режим доступа: https://blog.testlodge.com/how-to-write-a-good-bug-report/ (дата обращения: 19.11.2024).

2. Ben Lutkevich. Language modeling // TechTarget [Электронный ресурс]. Режим доступа: https://www.techtarget.com/searchenterpriseai/definition/language-modeling (дата обращения 20.11.2024).

3. Language Models, Explained: How GPT and Other Models Work // Altexsoft [Электронный ресурс]. Режим доступа: https://www.altexsoft.com/blog/language-models-gpt/ (дата обращения 20.11.2024).

4. Fawad Ali. GPT-1 to GPT-4: Each of OpenAI's GPT Models Explained and Compared // MakeUseOf [Электронный ресурс]. Режим доступа: https://www.makeuseof.com/gpt-models-explained-and-compared/ (дата обращения 21.11.2024).

5. Claude AI Overview // Anthropic [Электронный ресурс]. Режим доступа: https://www.anthropic.com/index/claude (дата обращения: 29.11.2024).

6. Описание API системы Intradesc // Intradesc [Электронный ресурс]. Режим доступа: https://intradesk.ru/api/ (дата обращения: 05.12.2024).