DOI 10.37882/2223-2966.2025.07.15

ИНТЕГРАЦИЯ ВНЕШНИХ ИСТОЧНИКОВ ДАННЫХ В СИСТЕМУ УЧЕТА ОТЧЕТНОСТИ ТЕЛЕКОММУНИКАЦИОННОЙ КОМПАНИИ: АРХИТЕКТУРА И РЕАЛИЗАЦИЯ

INTEGRATION OF EXTERNAL DATA SOURCES INTO THE ACCOUNTING SYSTEM OF A TELECOMMUNICATION COMPANY: ARCHITECTURE AND IMPLEMENTATION

I. Ilyin

Summary. To improve the completeness and integrity of data, telecommunications companies need to solve the problem of integrating various external data sources into internal accounting systems.
The article describes the implemented architecture of the integration module with the accounting system of technical maintenance reports in PJSC VimpelCom.
The analysis demonstrates the need for a variety of interfaces (REST API, mobile application, telegram bot, web interface) for effective work with various categories of users (company employees, employees of various contractors, other systems within the company).
The implementation of the integration module has improved the transparency of the maintenance process, reduced the time for data processing, and increased the reliability of interaction between systems.

Keywords: JSON, Kafka, REST API, Telegram bot, data integration, telecommunications, maintenance, token, reporting.

Ильин Илья Игоревич
Аспирант, ФГБОУ ВО «Московский государственный технологический университет Станкин»
hitsukey@yandex.ru

Аннотация. Для повышения полноты и целостности данных, телекоммуникационным компаниям требуется решить задачу интеграции различных внешних источников данных во внутренние системы учета.
В статье описана реализованная архитектура модуля интеграций с системой учета отчетности о техническом обслуживании в компании ПАО «Вымпелком».
Проведенный анализ демонстрирует необходимость многообразия интерфейсов (REST API, мобильное приложение, telegram-бот, веб-интерфейс) для эффективной работы с различными категориями пользователей (сотрудники компании, сотрудники различных подрядных организаций, другие системы внутри компании).
Внедрение модуля интеграций позволило обеспечить улучшение прозрачности процесса проведения технического обслуживания, сократить время на обработку данных, а также повысить надежность взаимодействия между системами.

Ключевые слова: JSON, Kafka, REST API, Telegram-бот, интеграция данных, телекоммуникации, техническое обслуживание, токен, учет отчетности.

Введение

Телекоммуникационные компании работают с большим количеством распределенных объектов и подрядных организаций, что требует постоянного обмена данными между различными системами учета, мобильными приложениями и сервисами автоматизации. Особенно остро стоит задача консолидации и учета отчетов о техническом обслуживании (ТО) базовых станций, которые формируются как вручную, так и автоматически. Для повышения прозрачности процессов, снижения количества ошибок и оптимизации документооборота необходимо внедрение решений, обеспечивающих прием данных из внешних источников, их проверку, маршрутизацию и регистрацию в системе учета отчетности.

Целью данной работы является проектирование и реализация архитектуры интеграции внешних источников данных в систему учета отчетности телекоммуникационной компании, а также внедрение механизма обмена данными с мобильными приложениями, чат-ботами и корпоративными платформами.

Научная значимость работы заключается в обобщении практики проектирования распределенной интеграционной архитектуры, устойчивой к изменениям в источниках данных и масштабируемой под различные интерфейсы взаимодействия. Практическая значимость состоит во внедрении решений, уже используемых в ПАО «Вымпелком».

Литературный обзор

Интеграция внешних источников данных в корпоративные учетные системы является ключевым направлением цифровой трансформации в сфере телекоммуникаций. Эффективность автоматизации отчетности зависит от выбора архитектурных решений, способов синхронизации данных и применяемых технологий обмена информацией между разнородными системами.

На примере автоматизированных производственных систем Аксенов К.А. и Спицина И.А. [1] исследуют практическую реализацию задач интеграции с учетом специфики сложных объектов. Это особенно актуально для внедрения модулей обработки отчетов, получаемых с различных базовых станций и объектов телекоммуникационной инфраструктуры.

Особенности цифровизации отчетности и её влияние на производственные и сервисные отрасли рассмотрены в работе Амановой Ай., Сахедова Я. и Мыратлыева Д. [2], где подчеркивается роль автоматизации как этапа перехода к цифровому управлению данными. Подобные преобразования становятся особенно актуальными для организаций с распределённой структурой и высокой интенсивностью документооборота, таких как телекоммуникационные компании.

В статье Багаутдинова К.Ш. подтверждается актуальность задачи интеграции систем. В своей работе он рассматривает методы интеграции информационных систем, архитектуры систем интеграции, механизмы отображения моделей данных, современные подходы к интеграции данных. Эти исследования могут лечь в основу разрабатываемых способов интеграции внешних источников данных в систему учета отчетности телекоммуникационной компании.

Белалова Г.А. [4] проводит классификацию методов интеграции и подчеркивает важность выбора адаптивной архитектуры при проектировании распределенных информационных систем, способных масштабироваться под нагрузку и изменяющиеся требования. Эти аспекты критичны для устойчивой обработки большого массива отчетных данных.

Иванюгин М.А. [5] рассматривает общие принципы взаимодействия между подсистемами в интеграционных архитектурах, акцентируя внимание на стандартах и протоколах обмена. Это имеет значение при реализации REST API и формировании гибкой модели прав доступа, как в системах сбора и контроля отчетности.

Особенности системной интеграции в транспортной логистике приведены у Искандерова Ю.М. [6], где подчеркивается роль согласованной структуры данных и мониторинга обмена. Это перекликается с задачами регистрации и контроля отправки/приема отчетов в распределенных учетных системах телекоммуникационного сектора.

Общие подходы к интеграции информационных систем и особенности их реализации в разнородных ИТ-ландшафтах рассмотрены в работе Карпова О.Э., Субботина С.А., Здирук К.К., Шишканова Д.В., Дьяченко П.С., Толпыгина А.С. и Стрельцова А.Н. [7]. Авторы делают акцент на практических аспектах внедрения интеграционных решений в крупной медицинской организации. Эти принципы могут быть адаптированы к нуждам телекоммуникационной отрасли при построении систем учета технического обслуживания (ТО).

Исторический и технологический обзор развития технологий интеграции информационных систем приводит Тёмкина Т.А. [8], выделяя эволюцию от централизованных решений к микросервисной архитектуре. Подобная архитектура обеспечивает модульность, отказоустойчивость и масштабируемость системы учета отчетности.

Практические аспекты построения интеграционных решений и управления правами доступа к ним раскрываются в работе Тимакина О.А. и Радзивон В. [9], что может быть использовано при проектировании API-интерфейсов для модулей взаимодействия с внешними приложениями (мобильными, ботами и др.).

В своей статье Хан А.О. и Никитин К.А. [10] рассматривают основные компоненты интегрированных информационных систем, а также их преимущества. Авторы заявляют, что такие системы предоставляют платформу для инноваций через внедрение передовых технологий, таких как искусственный интеллект, машинное обучение, большие данные и автоматизация процессов, которые могут открыть новые направления для роста и развития предприятия.

Таким образом, проведенный анализ литературы подтверждает высокую степень разработанности теоретических и практических основ интеграции информационных систем. Это создаёт основу для разработки специализированного решения — модуля интеграции внешних источников данных в систему учета отчетности о проведенном ТО в телекоммуникационной компании.

Материалы и методы

Часть пользователей системы проводит ТО и фиксирует собранные данные с помощью мобильного приложения. При завершении работ, приложение формирует JSON-объект с данными и отправляет его в Kafka. Модуль интеграций подписан на соответствующий топик в Kafka, из которого считывает сообщения. Полученный JSON и сопроводительная информация по запросу фиксируется в БД. Все полученные JSON-объекты обрабатываются и парсятся, а все полученные данные распределяются по таблицам БД. После записи в БД система формирует ответный JSON с результатом обработки и отправляет его обратно в Kafka, откуда мобильное приложение считывает сообщение и отображает результат пользователю во всплывающем уведомлении.

Для этой интеграции в БД разработана таблица BeetrackKafka, которая хранит в себе всю основную информацию по каждому запросу. Поля таблицы, их типы и описания представлены в таблице 1.

Таблица 1.

Структура таблицы БД BeetrackKafka

Поле	Тип	Описание
ID	int	Первичный ключ. Уникальный идентификатор записи в таблице.
Key	varchar(50)	Ключ запроса, идентифицирующий его в Kafka.
InsertTimestamp	timestamp	Дата и время добавления записи в таблицу.
RequestTimestamp	timestamp	Дата и время добавления запроса в Kafka.
RequestPayload	longtext	JSON с данными о проведенном ТО.
ResponsePayload	text	JSON с ответом от системы.
IsWorked	bit	Флаг, обработан ли запрос системой.
WorkedTimestamp	timestamp	Дата и время обработки запроса системой.
IsResponseSent	bit	Флаг, отправлен ли ответ системы в Kafka.
ResponseSentTimestamp	timestamp	Дата и время оправки ответа системы в Kafka.

Система считывает из запроса в Kafka ключ (фиксируется в Key), время помещения запроса в Kafka (фиксируется в RequestTimestamp), JSON с данными проведенного ТО (фиксируется в RequestPayload). При добавлении этой информации в таблицу BeetrackKafka, фиксируется дата и время (в InsertTimestamp). После обработки JSON и записи полученных данных в соответствующих таблицах, в BeetrackKafka фиксируется факт и временная отметка обработки запроса системой (IsWorked и WorkedTimestamp). Когда все данные получены и стоит соответствующая отметка, система отправляет сформированный ответ на запрос (ResponsePayload) в Kafka, а в таблице BeetrackKafka фиксируется факт и временная отметка отправки ответа в Kafka (IsResponseSent и ResponseSentTimestamp).

Рис.1. Схема таблиц БД для хранения данных из мобильного приложения

Важно отметить, что для идентификации в системе любого ТО, необходима связка параметров, состоящая из года проведения ТО, ERP, типа ТО, номера ТО в рамках года.

На стороне мобильного приложения проведение ТО реализовано таким образом, что создается запрос на проведение работ, затем создается задача на каждого члена группы. В процессе выполнения ТО члены группы фиксируют различные формы по одному или нескольким типам ТО.

В связи с этими ограничениями на стороне мобильного приложения, формируемый JSON-объект с данными имеет такую структуру, что после его обработки, данные можно распределить по 4 таблицам БД. Схема этих таблиц БД представлена на рисунке 1.

Таблица BeetrackRequest хранит данные о дате и времени, когда данные появились в этих таблицах, годе проведения ТО, ERP кода позиции, на которой проводилось ТО. Поля таблицы, их типы и описания представлены в таблице 2.

Таблица 2.

Структура таблицы БД BeetrackRequest

Поле	Тип	Описание
ID	varchar(50)	Первичный ключ. Уникальный идентификатор записи в таблице. В это поле записывается значение Key запроса, полученного из Kafka.
InsertTimestamp	timestamp	Дата и время добавления записи в таблицу.
Year	int	Год проведения ТО.
ERP	varchar(11)	ERP код позиции, на которой проводилось ТО. Состоит из 11 цифр.

Таблица BeetrackTask хранит ID записи из таблицы BeetrackRequest и данные об ответственных исполнителях (контактная информация и ФИО). Поля таблицы, их типы и описания представлены в таблице 3.

Таблица BeetrackForm хранит ID записи из таблицы BeetrackTask и данные о типе и номере проведенного ТО. Поля таблицы, их типы и описания представлены в таблице 4.

Таблица BeetrackItem хранит ID записи из таблицы BeetrackForm и данные о параметрах и полученных значениях, а также их типах. Поля таблицы, их типы и описания представлены в таблице 5.

В ходе работы разработан REST API для работы с системой учета. Для осуществления гибкой настройки прав доступа, каждому потребителю присваивается свой уникальный токен (уникальная последовательность символов). Благодаря этому токену можно однозначно идентифицировать источник запроса. Каждому токену предоставляются права на часть методов REST

Таблица 3.

Структура таблицы БД BeetrackTask

Поле	Тип	Описание
ID	bigint	Первичный ключ. Уникальный идентификатор записи в таблице.
BeetrackRequestID	varchar(50)	Уникальный идентификатор (поле ID) связанной записи из таблицы BeetrackRequest.
ResponsiblePhone	varchar(250)	Контактная информация ответственного исполнителя, проводившего ТО.
ResponsibleFullname	varchar(50)	ФИО ответственного исполнителя, проводившего ТО.

Таблица 4.

Структура таблицы БД BeetrackForm

Поле	Тип	Описание
ID	bigint	Первичный ключ. Уникальный идентификатор записи в таблице.
BeetrackTaskID	bigint	Уникальный идентификатор (поле ID) связанной записи из таблицы BeetrackTask.
ToTypeID	int	ID типа ТО из соответствующей таблицы.
ToNumber	int	Номер проведенного ТО в рамках года.

Таблица 5.

Структура таблицы БД BeetrackItem

Поле	Тип	Описание
ID	bigint	Первичный ключ. Уникальный идентификатор записи в таблице.
BeetrackFormID	bigint	Уникальный идентификатор (поле ID) связанной записи из таблицы BeetrackForm.
Name	varchar(500)	Наименование параметра. В дальнейшем используется, например, при сборке отчета.
Value	varchar(2500)	Значение параметра, полученное в ходе проведения ТО.
TypeID	int	ID типа параметра. Дата, целое число, строка, и т.п. Типы хранятся в отдельном справочнике.

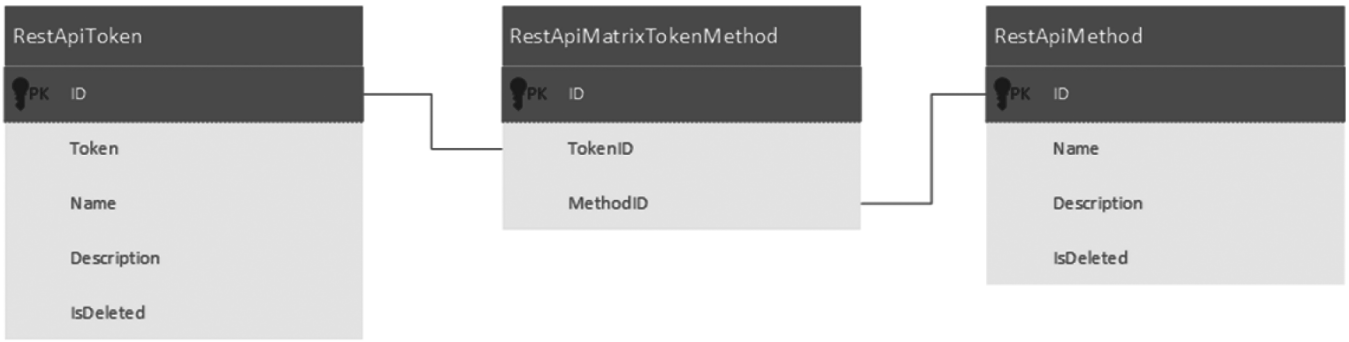

Рис. 2. Схема таблиц БД для управления правами доступа к методам REST API

API, в зависимости от согласования уровня прав. Для такого управления доступами к методам REST API, было разработано 3 таблицы БД. Схема этих таблиц представлена на рисунке 2.

Таблица RestApiToken хранит записи о токенах. Запись содержит уникальный ID, сам уникальный токен, наименование владельца токена (обычно какая-то система), описание (если требуется более подробное объяснение), флаг активности токена (удален он или нет). Поля таблицы, их типы и описания представлены в таблице 6.

Таблица 6.

Структура таблицы БД RestApiToken

Поле	Тип	Описание
ID	int	Первичный ключ. Уникальный идентификатор записи в таблице.
Token	varchar(50)	Уникальный токен, по которому предоставляется доступ.
Name	varchar(250)	Наименование системы, которой был предоставлен токен.
Description	varchar(1000)	Подробное описание системы или параметров интеграции.
IsDeleted	bit	Флаг, показывающий удалена запись или нет.

Таблица RestApiMethod хранит записи о методах. Запись содержит уникальный ID, название метода (который является его маркером), описание метода, флаг активности метода (удален он или нет). Поля таблицы, их типы и описания представлены в таблице 7.

Таблица RestApiMatrixTokenMethod хранит записи об активных связках токенов и методов. Наличие связки в этой таблице говорит о том, что у указанного токена есть права на доступ к указанному методу. Запись содержит уникальный ID, ID токена из таблицы RestApiToken, ID метода из таблицы RestApiMethod. Поля таблицы, их типы и описания представлены в таблице 8.

Таблица 7.

Структура таблицы БД RestApiMethod

Поле	Тип	Описание
ID	int	Первичный ключ. Уникальный идентификатор записи в таблице.
Name	varchar(250)	Маркер-название метода.
Description	varchar(1000)	Описание метода.
IsDeleted	bit	Флаг, показывающий удалена запись или нет.

Таблица 8.

Структура таблицы БД RestApiMatrixTokenMethod

Поле	Тип	Описание
ID	int	Первичный ключ. Уникальный идентификатор записи в таблице.
TokenID	int	Уникальный идентификатор токена.
MethodID	int	Уникальный идентификатор метода.

В качестве еще одного способа взаимодействия с системой учета, пользователям доступен telegram-бот, который взаимодействует с системой через REST API. Алгоритм взаимодействия пользователя с системой через telegram-бота показан на рисунках 3, 4.

Алгоритм работы:

1. Telegram-бот проверяет, авторизован ли пользователь (вызывается соответствующий метод API для проверки по номеру телефона).
2. Если пользователь не авторизован:

2.1. Telegram-бот отображает пользователю сообщение и кнопку «Авторизация».

2.2. Пользователь нажимает на кнопку «Авторизация».

2.3. Telegram-бот вызывает метод API для проверки УЗ пользователя. Проверяется, есть его УЗ в системе и есть ли у него доступ к работе с telegram-ботом.

2.4. Если пользователь найден и доступ есть:

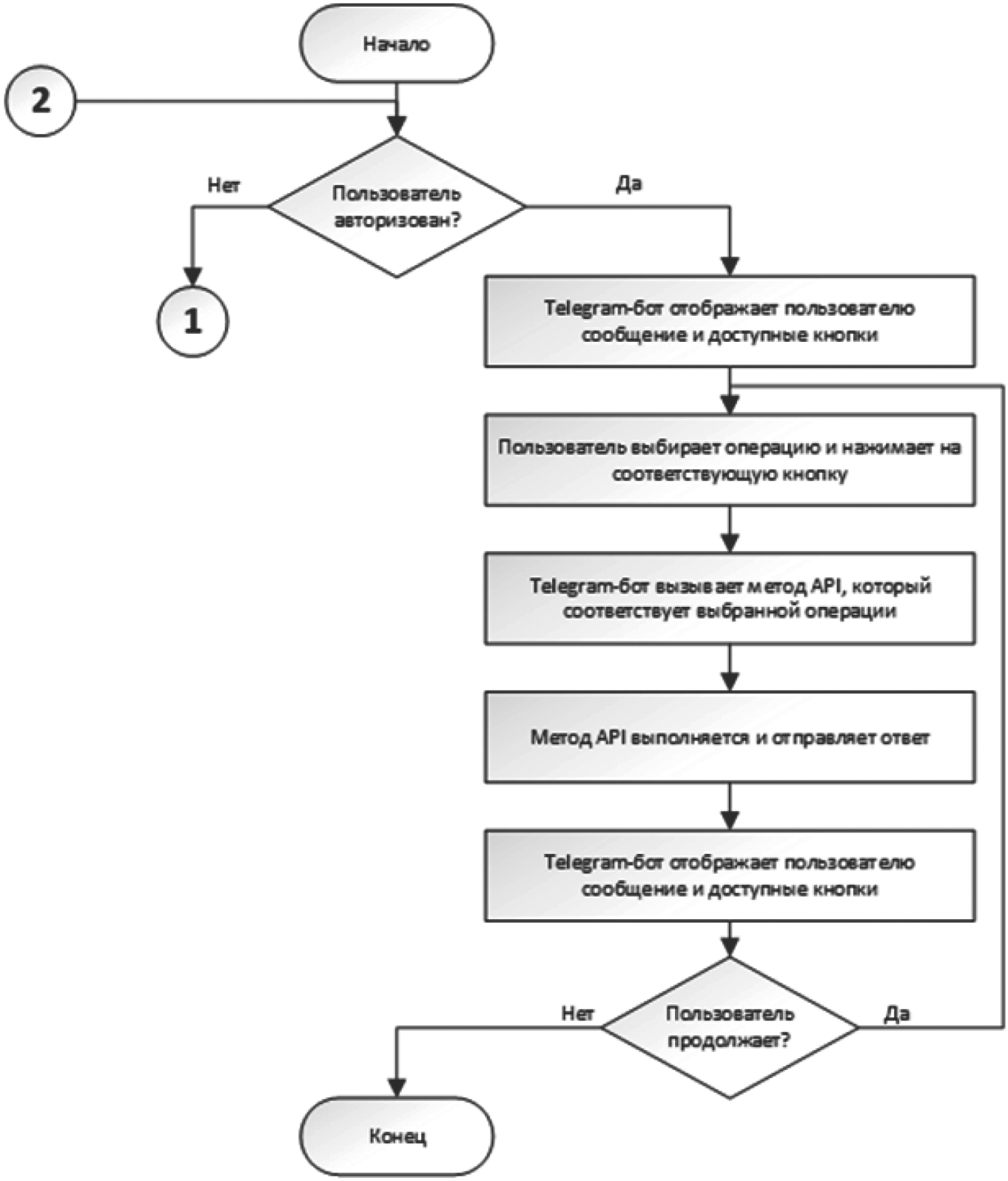

Рис. 3. Алгоритм работы с системой через telegram-бота — часть 1

2.4.1. Метод API возвращает сообщение об ошибке.
2.5. Еслипользовательненайденилиунегонетдоступа:
2.5.1. Метод API фиксирует в БД telegramID пользователя.
2.5.2. Метод API возвращает сообщение об успехе.
2.6. Telegram-бот отображает пользователю полученное сообщение.
3. Если пользователь авторизован:

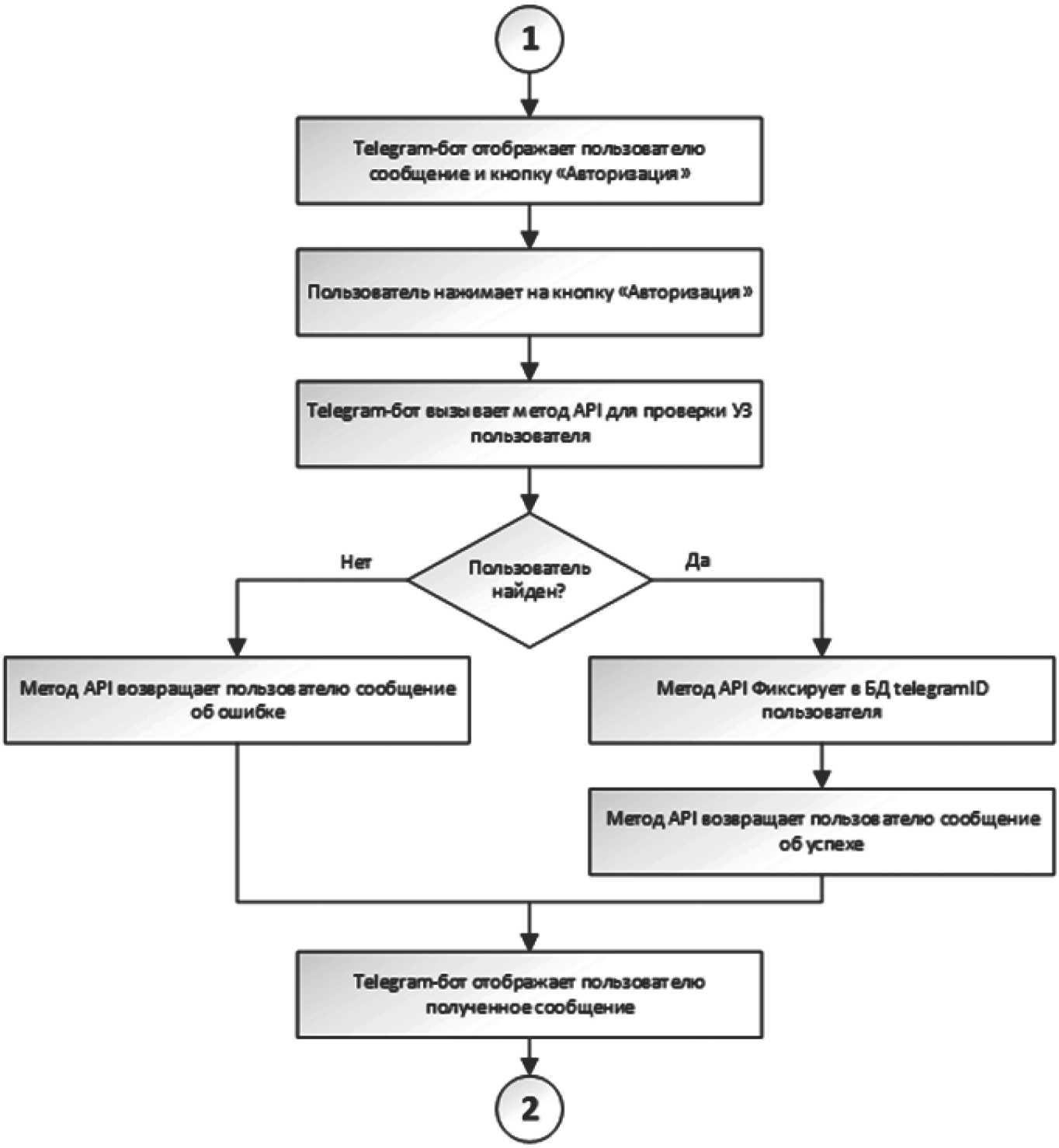

Рис. 4. Алгоритм работы с системой через telegram-бота — часть 2

3.1. Telegram-бот отображает пользователю сообщение и доступные кнопки.
3.2. Пользователь выбирает операцию и нажимает на соответствующую кнопку.
3.3. Telegram-бот вызывает метод API, который соответствует выбранной операции.
3.4. Метод API выполняется и отправляет результат в сторону telegram-бота.
3.5. Telegram-бот отображает пользователю полученный ответ и доступные кнопки.
4. Если пользователь завершил работу, то он выходит из чата с telegram-ботом. Есть опция сбросить авторизацию, если это требуется.

Результаты

В ходе работы был разработан модуль интеграции с системой. Внедрение разработанного модуля в ПАО «Вымпелком» дало следующие результаты:

- Успешно реализована интеграция с мобильным приложением. Более 3 000 операций ТО в месяц проходят через Kafka и обрабатываются без участия оператора.
- Разработан REST API, к которому подключены 5 внешних систем и 2 внутренних микросервиса.
- Telegram-бот используется более 80 сотрудниками подрядных организаций, выполняющими ТО в полевых условиях.
- Время регистрации отчета о ТО сократилось с 30 минут до менее чем 3 минут.
- Объем ручного ввода данных снизился на 60 %.
- Инциденты, связанные с несвоевременной передачей отчетов, сократились на 40 %.

Обсуждение

Использование распределенной архитектуры позволяет гибко расширять систему: подключать новых потребителей, масштабировать Kafka-топики, добавлять новые методы API. Механизм токенов обеспечивает безопасный и контролируемый доступ, а их иерархическая настройка прав упрощает администрирование. Telegram-бот обеспечивает доступ к системе даже в условиях ограниченного интернет-соединения, облегчая работу подрядчиков.

Главным ограничением является необходимость строгого соблюдения формата JSON и структуры сообщений, а также дополнительная нагрузка на команду поддержки при появлении новых потребителей.

Заключение

Реализация интеграции внешних источников данных с системой учета отчетности в ПАО «Вымпелком» позволила повысить скорость и качество регистрации данных о техническом обслуживании базовых станций. Разработка Kafka-интеграции с мобильным приложением, REST API с разграничением прав и Telegram-бота обеспечила многоуровневое и отказоустойчивое взаимодействие с различными потребителями. В будущем планируется расширение API, внедрение валидации входящих данных и формирование единой платформы мониторинга состояния всех подключенных систем. Полученные результаты демонстрируют высокую эффективность и масштабируемость предложенной архитектуры.

ЛИТЕРАТУРА

1. Аксенов К.А., Спицина И.А. Решение задачи интеграции информационных систем на примере автоматизированной системы выпуска металлургической продукции // ИВД. 2023. №6 (102). URL: https://cyberleninka.ru/article/n/reshenie-zadachi-integratsii-informatsionnyh-sistem-na-primere-avtomatizirovannoy-sistemy-vypuska-metallurgicheskoy-produktsii (дата обращения: 03.04.2025).
2. Аманова Ай., Сахедов Я., Мыратлыев Д. Модернизация информационных систем в условиях диджитальной трансформации // Вестник науки. 2024. №9 (78). С. 296–300.
3. Багаутдинов К.Ш. Методы интеграции информационных систем на основе универсального анализатора онлайн-информации // ВК. 2019. №3 (35). С. 52–60. DOI: 10.34822/1999-7604-2019-3-52-60
4. Белалова Г.А. Анализ методов интеграции информационных систем // Цифровые модели и решения. 2023. №3. С. 61–68. DOI: 10.29141/2949-477X-2023-2-3-5
5. Иванюгин М.А. Интеграция информационных систем // Международный журнал гуманитарных и естественных наук. 2020. №5–1. С. 80–83. DOI: 10.24411/2500-1000-2020-10455
6. Искандеров Ю.М. Интеграция информационных ресурсов для управления транспортными процессами: системные аспекты // SAEC. 2021. №3. С. 254–260. DOI: 10.18720/SPBPU/2/id21-375
7. Карпов О.Э., Субботин С.А., Здирук К.К., Шишканов Д.В., Дьяченко П.С., Толпыгин А.С., Стрельцов А.Н. Интеграция с внешними информационными системами. Особенности многопрофильного медицинского учреждения // Вестник Национального медико-хирургического Центра им. Н.И. Пирогова. 2018. №4. С. 4–9. DOI: 10.25881/BPNMSC.2018.73.80.001
8. Тёмкина Т.А. Эволюция технологий интеграции информационных систем // T-Comm. 2011. №7. С. 151–155.
9. Тимакин О.А., Радзивон В. Описание интеграционных решений информационной системы иособенности ее использования // Евразийский научный журнал. 2015. №12. С. 298–303.
10. Хан А.О., Никитин К.А. Интегрированные информационные системы: инновационный путь к цифровой трансформации // Вестник науки. 2024. №5 (74). С. 1569–1577.

Журнал «Современная наука: актуальные проблемы теории и практики»