УДК 004.41:004.738.5

Мушкалов Е.А.
студент 4 курса направления 01.03.02 «Прикладная математика и информатика»,
Институт компьютерных наук и технологий, ПГНИУ
Россия, г. Пермь
Минасян Т.А.
студент 4 курса направления 01.03.02 «Прикладная математика и информатика»,
Институт компьютерных наук и технологий, ПГНИУ
Россия, г. Пермь

АРХИТЕКТУРНАЯ МОДЕЛЬ СОБЫТИЙНО-ОРИЕНТИРОВАННОГО ФРЕЙМВОРКА ИНТЕГРАЦИИ КОРПОРАТИВНОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ С TELEGRAM BOT API

Аннотация: в статье рассматривается задача проектирования фреймворка интеграции корпоративной информационной системы на платформе «1С:Предприятие» с мессенджером Telegram. Актуальность работы связана с тем, что при добавлении прикладных сценариев Telegram-бота повторяются одни и те же технические механизмы: прием обновлений, маршрутизация команд, хранение состояния диалога, формирование inline-интерфейса и отправка ответов. Цель исследования состоит в разработке архитектурной модели, в которой универсальное ядро взаимодействия с Telegram Bot API отделено от прикладной бизнес-логики. В качестве методов использованы архитектурная декомпозиция, модель конечного автомата для описания многоэтапного диалога и анализ средств платформы «1С:Предприятие». Результатом исследования

является событийно-ориентированная модель фреймворка, алгоритм обработки входящего обновления и принципы расширения системы новыми прикладными сценариями.

Ключевые слова: *корпоративная информационная система, «1С:Предприятие», Telegram Bot API, фреймворк интеграции, webhook, конечный автомат, архитектурная модель.*

Mushkalov E.A.
4th-year student, Applied Mathematics and Computer Science,
Institute of Computer Science and Technology, Perm State University
Russia, Perm
Minasyan T.A.
4th-year student, Applied Mathematics and Computer Science,
Institute of Computer Science and Technology, Perm State University
Russia, Perm

ARCHITECTURAL MODEL OF AN EVENT-ORIENTED FRAMEWORK FOR INTEGRATING A CORPORATE INFORMATION SYSTEM WITH TELEGRAM BOT API

Abstract: the article examines the design of an integration framework connecting a corporate information system based on the 1C:Enterprise platform with Telegram messenger. The relevance of the study is determined by the repeated implementation of the same technical mechanisms when new Telegram bot scenarios are added: receiving updates, routing commands, storing dialog state, building inline interfaces, and sending responses. The purpose of the study is to develop an architectural model in which the universal interaction core with Telegram Bot API is separated from applied business logic. The research methods include architectural decomposition, finite-state dialog

modeling, and analysis of standard 1C:Enterprise mechanisms. The result is an event-oriented framework model, an algorithm for processing incoming updates, and principles for extending the system with new applied scenarios.
Keywords: *corporate information system, 1C:Enterprise, Telegram Bot API, integration framework, webhook, finite-state machine, architectural model.*

Введение

Корпоративные информационные системы постепенно перестают быть только рабочим местом в отдельном клиентском приложении. Во многих организациях пользователю требуется быстро получить уведомление, передать данные, поставить задачу или запросить отчет без полноценного открытия системы. Поэтому в качестве дополнительного канала взаимодействия все чаще используются мессенджеры, а Telegram-бот становится интерфейсом между пользователем и внутренней учетной или управленческой системой [1].

Для платформы «1С:Предприятие» такая интеграция имеет практическое значение, поскольку типовые и нетиповые конфигурации уже содержат сведения о задачах, пользователях, организациях, выполненных работах и регламентных действиях. Telegram Bot API позволяет связать эти данные с мобильным каналом взаимодействия: принять сообщение через webhook, обработать нажатие inline-кнопки, отправить текст, документ или уведомление [2, 3]. Однако сам факт наличия программного интерфейса не решает архитектурную проблему построения расширяемого бота.

При разработке нескольких сценариев в одном Telegram-боте быстро обнаруживается повторяемость технических операций. Независимо от того, создает ли пользователь задачу, выбирает период отчета, прикрепляет файл или подтверждает выполнение действия, серверная часть должна получить событие, определить пользователя, восстановить

текущий шаг диалога, выбрать обработчик, сформировать ответ и сохранить новое состояние. Если эти действия реализуются отдельно в каждом сценарии, код становится моноблочным, усложняется сопровождение и возрастает риск расхождений в поведении разных разделов бота.

Наиболее устойчивым решением является выделение общего программного слоя, который обеспечивает взаимодействие с Telegram Bot API и предоставляет прикладным сценариям единые функции приема, маршрутизации, хранения состояния и отправки сообщений. Такой слой можно рассматривать как фреймворк интеграции. Он не заменяет бизнес-логику корпоративной системы, а задает правила ее подключения к внешнему коммуникационному каналу.

Цель настоящей статьи – представить архитектурную модель событийно-ориентированного фреймворка интеграции корпоративной информационной системы на платформе «1С:Предприятие» с Telegram Bot API. Для достижения цели необходимо выделить повторяющиеся механизмы взаимодействия, определить состав универсального ядра, описать модель управления состояниями диалога, предложить алгоритм обработки входящего обновления и показать применимость решения на прикладных сценариях учета задач и отчетности по затраченному времени.

Методы и исследования

Методической основой исследования выступает архитектурная декомпозиция интеграционного решения [4]. Вместо рассмотрения Telegram-бота как набора отдельных команд предлагается рассматривать его как поток событий, поступающих из внешнего канала в корпоративную информационную систему. К событиям относятся текстовые сообщения, команды, callback-запросы от inline-кнопок и переданные пользователем файлы. Каждое событие должно быть обработано с учетом текущего состояния пользовательского диалога.

Вторым методическим основанием является разделение универсальной и прикладной частей. Универсальная часть не должна зависеть от конкретной сущности предметной области, например от документа «Задача» или отчета по времени. Ее назначение состоит в обработке транспортного уровня, маршрутизации и сервисного состояния. Прикладная часть, напротив, реализует конкретные действия корпоративной системы: создание документа, получение данных из регистров, формирование отчета или запись результата в базу.

Для описания многоэтапных пользовательских сценариев используется модель конечного автомата [5]. Состоянием автомата является стадия диалога пользователя, входным событием – очередное сообщение или нажатие кнопки, а функцией перехода – обработчик прикладного сценария. Дополнительные параметры сценария образуют расширенное состояние: например, выбранный пользователь, период отчета, дата начала задачи или текстовое описание. Такая модель удобна для Telegram-бота, поскольку пользовательский диалог физически разделен на отдельные сообщения.

Технологическая часть исследования опирается на штатные механизмы платформы «1С:Предприятие». Общий модуль используется как место размещения серверных процедур фреймворка, регистры сведений – как хранилище текущих и отложенных состояний, HTTP-сервис – как точка приема webhook-запросов, фоновые задания – как способ асинхронной обработки обновления, а регламентные задания – как механизм периодических уведомлений и обработки отложенных запросов [6, 7, 8]. Отказ от внешнего сервиса-посредника делает архитектуру более управляемой для нетиповой корпоративной конфигурации.

В рамках исследования также учитываются ограничения Telegram Bot API. Webhook-режим предполагает быстрый ответ сервера на входящее обновление, поэтому длительная прикладная обработка не

должна выполняться непосредственно внутри HTTP-запроса. Кроме того, интерактивный интерфейс бота строится через inline-клавиатуры, а значит, архитектура должна включать единый механизм формирования callback_data и последующего сопоставления этого значения с обработчиком сценария [2, 3].

Результаты оригинального авторского исследования

В результате исследования предложена архитектурная модель фреймворка, основанная на событийном принципе. Входящее обновление Telegram рассматривается как событие, которое проходит через последовательность универсальных этапов: прием, классификация, идентификация пользователя, восстановление состояния, маршрутизация, выполнение обработчика и сохранение результата. Прикладной сценарий подключается только на этапе выполнения бизнес-действия и не реализует низкоуровневую работу с Telegram Bot API.

Ключевым элементом модели является универсальное ядро. Оно объединяет процедуры приема входящих событий, отправки сообщений и файлов, построения inline-клавиатур, регистрации пользователя, журналирования, управления текущими и отложенными запросами. Ядро предоставляет прикладным сценариям единый программный интерфейс. За счет этого новый сценарий добавляется не путем копирования готовой команды, а через регистрацию обработчика и описание его стадий.

Состав основных компонентов универсального ядра представлен в таблице 1.

Таблица 1 – Основные компоненты универсального ядра фреймворка

Компонент	**Назначение**	**Реализация на платформе «1С:Предприятие»**
Прием событий	Получение webhook-запроса, выделение типа обновления и передача обработки в фоновое задание.	HTTP-сервис, фоновое задание, общий модуль.
Маршрутизация	Сопоставление команды, callback_data или текущего	Таблица команд, динамический вызов

Компонент	Назначение	Реализация на платформе «1С:Предприятие»
	состояния с процедурой-обработчиком.	обработчика, общий модуль.
Состояния диалога	Хранение текущей стадии сценария и накопленных параметров между сообщениями пользователя.	Регистр сведений текущих запросов, структура данных сценария.
Интерфейс и отправка	Формирование inline-клавиатур, текстовых сообщений, документов и уведомлений.	Функции формирования reply_markup и отправки запросов к Telegram Bot API.
Сервисные процессы	Журналирование, хранение последнего сообщения, обработка отложенных запросов и напоминаний.	Регистры сведений, регламентные задания, очередь оповещений.

Алгоритм обработки входящего обновления строится следующим образом. Сначала HTTP-сервис принимает POST-запрос от Telegram и извлекает из него служебные данные. Затем определяется бот, для которого получено обновление, и структура сообщения преобразуется во внутреннее представление платформы. Основная обработка запускается в фоновом задании, что позволяет быстро вернуть HTTP-ответ и не нарушать требования webhook-режима. После этого универсальное ядро классифицирует событие: текстовое сообщение передается в обработчик команд, callback-запрос – в обработчик inline-команд, а файл – в механизм приема и последующей прикладной обработки.

Маршрутизация имеет два уровня. Первый уровень связан с командами главного меню: для них заранее задано отображаемое название, техническое имя и процедура-обработчик. Второй уровень используется внутри многошагового сценария. Если у пользователя уже есть активный текущий запрос, фреймворк восстанавливает имя сценария, номер стадии и накопленные данные, после чего вызывает соответствующую процедуру. Такой подход позволяет отделить общий диспетчер от конкретного перечня прикладных сценариев.

Управление состоянием диалога реализуется через запись текущего запроса. В записи хранится идентификатор пользователя Telegram, имя бота, имя команды, номер стадии, структура накопленных данных, время постановки состояния и время жизни запроса. Поле стадии задает текущее состояние конечного автомата, а структура данных позволяет каждому сценарию хранить собственный набор параметров без изменения универсальной схемы хранения. При завершении сценария запись текущего запроса удаляется, и следующие сообщения пользователя снова интерпретируются как новые команды.

Отдельным результатом является модель отложенных запросов. Она применяется в ситуациях, когда сценарий должен продолжиться позже: например, при напоминании, подтверждении действия или ожидании ответа в течение ограниченного времени. Отложенный запрос хранит тот же контекст, что и текущий диалог, но дополнительно содержит идентификатор будущего действия и сведения для отправки сообщения. При нажатии пользователем inline-кнопки фреймворк восстанавливает сохраненный контекст и передает управление обычному обработчику сценария.

Практическая проверка модели выполнена на двух типах прикладных сценариев. Первый сценарий связан с созданием задачи на себя через Telegram-бота. В нем пользователь последовательно выбирает вариант создания задачи, вводит цель, уточняет параметры и подтверждает режим старта. На финальном шаге прикладной обработчик создает документ в информационной системе. Универсальное ядро при этом обеспечивает прием сообщений, сохранение стадий, формирование inline-клавиатур и отправку подтверждения.

Второй сценарий связан с формированием отчета по затраченному времени. Пользователь выбирает команду отчета и период, после чего корпоративная система формирует результат и отправляет его через

Telegram в виде PDF-документа. Этот сценарий показывает, что фреймворк применим не только к операциям записи данных, но и к операциям чтения, компоновки отчетного результата и доставки файла пользователю.

Сравнение двух сценариев демонстрирует главное свойство предложенной модели: различается только прикладная логика, тогда как транспорт, состояние, интерфейс и отправка результата выполняются едиными средствами ядра. Это снижает объем повторяющегося кода и делает добавление новых команд предсказуемым. Разработчику нового сценария достаточно описать последовательность стадий, зарегистрировать команду или вложенный callback-вариант и использовать функции ядра для взаимодействия с пользователем.

Предложенная архитектура допускает постепенное развитие. Минимальный вариант может включать прием webhook, регистр текущих состояний, маршрутизацию команд и отправку сообщений. Более полный вариант добавляет журналирование, отложенные запросы, обработку файлов, плановые уведомления и единый механизм обновления последнего сообщения. Такое поэтапное внедрение особенно важно для нетиповых корпоративных конфигураций, где интеграция развивается вместе с реальными бизнес-процессами организации.

Ограничением модели является необходимость дисциплины при разработке прикладных сценариев. Если обработчики начинают напрямую обращаться к Telegram Bot API или самостоятельно хранить состояние, архитектурное разделение нарушается. Поэтому фреймворк должен сопровождаться соглашениями для разработчиков: единая сигнатура процедур, обязательное использование текущего запроса, централизованное формирование inline-клавиатур и завершение сценария через удаление сохраненного состояния.

Заключение

В статье предложена архитектурная модель событийно-ориентированного фреймворка интеграции корпоративной информационной системы на платформе «1С:Предприятие» с Telegram Bot API. Основная идея состоит в разделении универсального ядра и прикладных сценариев. Ядро отвечает за прием событий, маршрутизацию, хранение состояния, формирование интерфейса, отправку сообщений и сервисную обработку, а прикладные сценарии реализуют конкретные действия в предметной области корпоративной системы.

Сформирована модель управления многоэтапным диалогом на основе конечного автомата. Текущая стадия, имя команды и накопленные данные сохраняются в служебном хранилище, что позволяет восстанавливать контекст пользователя при каждом новом сообщении. Для сценариев, требующих продолжения в будущем, предложен механизм отложенных запросов, сохраняющий контекст будущего взаимодействия и связывающий его с inline-кнопками или регламентной обработкой.

Практическая ценность результата заключается в снижении дублирования кода при развитии Telegram-бота и в упрощении добавления новых сценариев. Апробация на сценариях создания задачи и формирования отчета по затраченному времени показывает, что одна и та же архитектурная основа подходит как для операций записи данных, так и для формирования отчетной информации с последующей отправкой файла пользователю.

Дальнейшее развитие модели может быть связано с формализацией шаблонов прикладных сценариев, расширением средств диагностики пользовательских диалогов, добавлением контроля прав доступа на уровне маршрутизации и разработкой набора типовых компонентов для часто используемых действий: выбора периода, прикрепления файла, подтверждения операции и отправки уведомлений группе пользователей.

Использованные источники

1. Волик М. В. Корпоративные информационные системы на базе 1С:Предприятие 8: учебное пособие / М. В. Волик. – Москва: Прометей, 2020. – 102 с. – ISBN 978-5-907244-00-9.
2. Telegram Bot API [Электронный ресурс] // Telegram: официальный сайт. – URL: https://core.telegram.org/bots/api (дата обращения: 04.06.2026).
3. Telegram Bots Webhooks [Электронный ресурс] // Telegram: официальный сайт. – URL: https://core.telegram.org/bots/webhooks (дата обращения: 04.06.2026).
4. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес; пер. с англ. – Санкт-Петербург: Питер, 2015. – 368 с. – ISBN 978-5-496-00389-6.
5. Хопкрофт Дж. Введение в теорию автоматов, языков и вычислений / Дж. Хопкрофт, Р. Мотвани, Дж. Ульман; пер. с англ. – 2-е изд. – Москва: Вильямс, 2002. – 528 с.
6. Радченко М. Г. 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы / М. Г. Радченко, Е. Ю. Хрусталева. – 3-е изд. – Москва: 1С-Паблишинг, 2023. – 957 с.
7. Хрусталева Е. Ю. Технологии интеграции «1С:Предприятия 8.3» / Е. Ю. Хрусталева. – 2-е изд., стер. – Москва: 1С-Паблишинг, 2021. – 498 с.
8. 1С:Предприятие 8.3. Руководство разработчика [Электронный ресурс] // 1С:ИТС: официальный сайт. – URL: https://its.1c.ru/ (дата обращения: 04.06.2026).