УДК 004.4’2:004.55

Романенко И. П.,
магистрант
ФГБОУ ВО «Кубанский государственный университет»
Россия, г. Краснодар

АРХИТЕКТУРА И ПРОЕКТИРОВАНИЕ ИНТЕГРИРОВАННОЙ СИСТЕМЫ ОПТИМИЗАЦИИ ПРИЕМНОЙ КАМПАНИИ УНИВЕРСИТЕТА НА БАЗЕ TELEGRAM-БОТА (НА ПРИМЕРЕ КУБАНСКОГО ГОСУДАРСТВЕННОГО УНИВЕРСИТЕТА)

Аннотация: В работе рассмотрена проблема автоматизации взаимодействия абитуриентов и приемной комиссии университета в условиях ежегодного роста числа заявлений и цифровизации коммуникаций. На примере Telegram-бота для Кубанского государственного университета проанализированы возможности и ограничения Telegram Bot API, обоснован выбор технологического стека (Python, aiogram, SQLite) и предложена модульная событийно-ориентированная архитектура системы. Архитектура включает интеграционный слой (обработка обновлений через Dispatcher и Router aiogram), слой управления состояниями (встроенная FSM) и логический слой функциональных модулей (регистрация с созданием личного кабинета, информационная поддержка FAQ, навигация и перенаправление). Описаны механизмы персистентного хранения данных в SQLite с параллельным логированием, обеспечение соответствия требованиям Федерального закона № 152-ФЗ, а также принципы разделения ответственности, позволяющие обеспечить надежность многошаговых сценариев и удобство расширения системы. Сформулированы архитектурные решения, которые могут быть перенесены на другие высшие учебные заведения.

***Ключевые слова*:** Telegram-бот, aiogram, FSM, SQLite, Long Polling, Webhook, приемная кампания, автоматизация, вуз, абитуриент, личный кабинет.

Введение

Современные высшие учебные заведения сталкиваются с ежегодным ростом числа абитуриентов и необходимостью оперативного предоставления информации в условиях цифровизации. Традиционные каналы связи — телефонные горячие линии, личные обращения и даже портал «Госуслуги» — в пиковые периоды приемной кампании создают значительную нагрузку на сотрудников приемных комиссий, приводят к задержкам ответов и снижению удовлетворенности абитуриентов. По данным приемной кампании Кубанского государственного университета за 2020–2024 годы, количество поданных заявлений стабильно превышает 89 тысяч ежегодно, при этом доля использования личного кабинета абитуриента остается крайне низкой (2,7 %). Почти треть заявлений поступает из других регионов РФ, что усиливает территориальные барьеры и потребность в круглосуточном, привычном для молодежи канале коммуникации.

Telegram как платформа получил широкое распространение среди молодежи благодаря открытости, кроссплатформенности и популярности мессенджера (Краснодарский край занимает 9-е место по числу пользователей по данным ПАО «МТС»). Существующие аналоги в российских вузах (боты НИУ МГСУ, ЮФУ и др.) демонстрируют эффективность автоматизации FAQ, отслеживания сроков и сбора данных, однако вопросы построения целостной модульной архитектуры Telegram-бота, способного обеспечить надежную регистрацию с созданием личного кабинета, масштабируемость и соответствие законодательству о персональных данных, остаются недостаточно проработанными.

Цель данной статьи — предложить и обосновать архитектурный подход к созданию интегрированной системы на базе Telegram-бота для оптимизации приемной кампании университета на примере Кубанского государственного

университета, а также описать ключевые технические решения, обеспечивающие устойчивость, производительность, защиту данных и удобство сопровождения.

Анализ среды исполнения виджетов amoCRM

Прежде чем проектировать архитектуру, необходимо проанализировать условия работы системы. Telegram предоставляет формализованную модель через Bot API — набор методов, реализуемых через HTTPS-запросы для отправки/получения сообщений, управления клавиатурами и обработки данных о пользователях и чатах. Центральными элементами являются:

• **BotFather** — официальный бот для создания и настройки бота, генерации уникального токена аутентификации, описания, фото и команд. • **Long Polling и Webhook** — технологии получения обновлений. Long Polling проще в реализации на этапе разработки и тестирования (постоянные запросы getUpdates с открытым туннелем), Webhook обеспечивает минимальную задержку в продакшене (требует публичного HTTPS-URL). • **Клавиатуры** — ReplyKeyboardMarkup (постоянная клавиатура под чатом) и InlineKeyboardMarkup (кнопки под конкретным сообщением) для интерактивного взаимодействия. • **Базы данных** — необходимы для сохранения контекста диалогов, данных пользователей и истории. Для MVP оптимальна легковесная встраиваемая SQLite (один файл .db, нулевая конфигурация, высокая производительность чтения).

Из этого вытекают архитектурные ограничения и возможности. Бот не контролирует весь интерфейс пользователя — взаимодействие происходит через сообщения и кнопки в мессенджере. Получение и обработка обновлений должны быть асинхронными для параллельной работы с множеством пользователей. Многошаговые сценарии (например, регистрация) требуют сохранения контекста между сообщениями. Интерфейс должен быть интуитивно понятным, быстрым и соответствовать ожиданиям молодежи (мгновенность, минимум действий).

Сравнительный анализ платформ показал преимущество Telegram: богатый Bot API, встроенная поддержка FSM в современных фреймворках, высокая популярность среди целевой аудитории. VKontakte и MAX уступают по гибкости API и удобству реализации сложных диалогов, хотя соответствуют требованиям суверенитета данных. Психологические аспекты (иллюзия конфиденциальности, мгновенная коммуникация, возможность самовыражения через стикеры и группы) дополнительно подтверждают выбор Telegram как привычного и комфортного канала для абитуриентов поколения Z и Alpha.

Требования к архитектуре интегрируемой системы на базе Telegram-бота

С учётом сказанного можно сформулировать ключевые требования к архитектуре разрабатываемой системы:

1. Совместимость с моделью Telegram Bot API и современными фреймворками (aiogram). Система должна естественно вписываться в жизненный цикл обновлений, использовать асинхронную обработку и не конфликтовать с ограничениями мессенджера.
2. Разделение ответственности. Интеграционный слой (обработка обновлений), слой управления состояниями и данными, а также логический слой функциональных модулей должны быть изолированы, чтобы изменение одного из них не вызывало каскадных правок.
3. Производительность и асинхронность. Система должна эффективно обрабатывать параллельные обновления от множества пользователей в пиковые периоды кампании без блокировки основного потока (использование asyncio и event-driven подхода).
4. Устойчивость к многошаговым сценариям и неоднородным данным. Архитектура должна предусматривать надежное управление контекстом диалога (FSM) и персистентное хранение данных с валидацией на каждом этапе, чтобы избежать неполных записей и ошибок ввода.

5. Соответствие законодательству о персональных данных. Хранение ФИО, контактов и Telegram ID должно осуществляться с соблюдением требований Федерального закона № 152-ФЗ (локальное хранение, ограничение доступа, отсутствие передачи сторонним сервисам).
6. Удобство сопровождения, расширения и масштабируемости. Добавление новых модулей (уведомления, интеграция с CRM) или перенос системы на другой вуз не должны требовать переписывания всей архитектуры. Модульность и событийно-ориентированный подход обеспечивают независимое развитие компонентов.

Предлагаемая архитектура

Опираясь на перечисленные требования, архитектура системы спроектирована как многоуровневая событийно-ориентированная структура на базе клиент-серверной модели. Пользователь взаимодействует через приложение Telegram; все события передаются на сервер разработчика через Bot API; обработка логики, работа с данными и формирование ответов происходят централизованно. В ней выделяются три основных горизонтальных слоя.

1. **Интеграционный слой** отвечает за все взаимодействия с внешним миром: прием обновлений (сообщения, команды /start, /help, callback-запросы), их распределение через Dispatcher и Router aiogram, настройку бота через BotFather (токен, описание, команды). Для разработки и тестирования используется Long Polling; в продакшене возможен переход на Webhook. Слой инкапсулирует работу с Bot API, обеспечивая единый транспортный уровень.
2. **Слой обработки данных и состояний** выполняет управление контекстом диалогов и персистентное хранение. Здесь реализована встроенная FSM aiogram (StatesGroup с последовательными состояниями), временное сохранение данных в FSMContext на каждом шаге и финальная запись в SQLite только после успешного завершения сценария. Дополнительно

ведется параллельное логирование в текстовый файл для отладки. Этот слой обеспечивает целостность данных и позволяет пользователю прерывать процесс без сохранения неполной информации.

3. **Логический слой модулей** строится на независимых функциональных блоках, которые получают уже подготовленные данные и не содержат собственной логики обработки обновлений. Модуль регистрации и создания личного кабинета реализует многошаговый сбор ФИО, email и телефона через FSM с сохранением в таблицу users базы SQLite. Модуль информационной поддержки FAQ предоставляет интерактивное меню на основе ReplyKeyboardMarkup с быстрыми ответами на типовые вопросы (документы, сроки). Модуль навигации и перенаправления обеспечивает мгновенный переход на сайт университета и предоставление контактов горячей линии через простые обработчики с фильтрами F.text.lower().

Дополнительно выделяются сквозные сервисы: менеджер базы данных SQLite (CRUD-операции, инициализация таблиц), система централизованного логирования (модуль logging Python с уровнями DEBUG–CRITICAL и оберткой try-except всех обработчиков) и менеджер состояний FSM. Эти сервисы используются на разных уровнях, но их логика централизована.

Такое разделение позволяет свести связанность между частями системы к минимуму. Замена Bot API, доработка FSM-логики или добавление нового модуля затрагивают лишь соответствующий слой. На рисунке 1 представлена схема предложенной архитектуры.

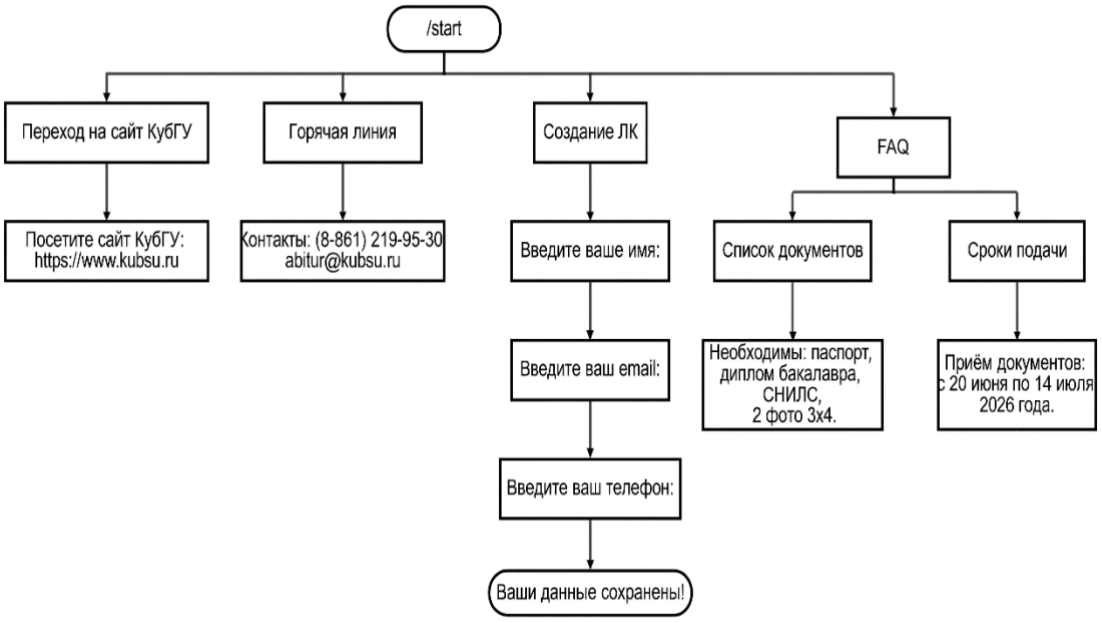

Рисунок 1 – Схема предложенной архитектуры.

Интеграционный слой строится на принципе «единого транспортного уровня». Все обновления принимаются Dispatcher’ом aiogram и распределяются между Router’ами и конкретными обработчиками (@dp.message, @dp.callback_query с фильтрами). Благодаря этому модули не знают деталей протокола Bot API.

Критически важной задачей является реализация многошаговой регистрации. Без FSM привязка к одному формату сообщений сделала бы модуль хрупким. В архитектуре используется встроенная FSM aiogram: на каждом состоянии данные временно сохраняются в контексте, валидируются и только после завершения всех этапов записываются в SQLite и лог-файл. Такой подход гарантирует целостность данных и удобство пользователя.

Значительная часть данных (ответы FAQ, контакты, ссылки) редко меняется. Повторная загрузка или сложные запросы создавали бы избыточную нагрузку. Для решения проблемы в архитектуру интегрирована легковесная SQLite с индексацией и параллельным текстовым логированием. При необходимости система

предусматривает постепенный переход к более масштабируемым СУБД (PostgreSQL) и шифрованию (SQLCipher).

Заключение

В статье представлен архитектурный подход к построению интегрированной системы оптимизации приемной кампании университета на базе Telegram-бота, рассмотренный на примере Кубанского государственного университета. Предложенная модульная событийно-ориентированная структура с разделением на интеграционный слой, слой управления состояниями и данными, а также логический слой функциональных модулей обеспечивает устойчивость к изменениям Telegram Bot API, упрощает сопровождение кода и позволяет независимо развивать отдельные сценарии взаимодействия.

Ключевыми техническими решениями, повышающими практическую ценность архитектуры, являются: выбор асинхронного фреймворка aiogram с встроенной поддержкой FSM, интеграция SQLite для надежного хранения данных личного кабинета с параллельным логированием, модульная реализация FAQ и навигации на основе клавиатур, а также меры по защите персональных данных в соответствии с Федеральным законом № 152-ФЗ. Данные решения позволяют снизить нагрузку на сотрудников приемной комиссии, обеспечить мгновенный доступ абитуриентов к информации в привычном мессенджере и создать основу для дальнейшего развития (персонализированные уведомления, интеграция с внутренними системами вуза). Принципы архитектуры могут быть использованы в качестве методической основы при разработке аналогичных систем для других высших учебных заведений и приемных кампаний.

Список литературы:

1. Telegram. Bot API Reference [Электронный ресурс] // Официальная документация Telegram. — URL: https://core.telegram.org/bots/api (дата обращения: 20.05.2026).

2. aiogram. Official Documentation [Электронный ресурс] // aiogram framework. — URL: https://docs.aiogram.dev/en/v3.28.2/ (дата обращения: 20.05.2026).
3. Telegram. BotFather [Электронный ресурс] // Официальная документация Telegram. — URL: https://core.telegram.org/bots#botfather (дата обращения: 20.05.2026).
4. Федеральный закон от 27.07.2006 № 152-ФЗ «О персональных данных» [Электронный ресурс] // КонсультантПлюс. — URL: https://www.consultant.ru/document/cons_doc_LAW_61801/ (дата обращения: 22.05.2026).
5. Python Software Foundation. asyncio — Asynchronous I/O [Электронный ресурс] // Python Documentation. — URL: https://docs.python.org/3/library/asyncio.html (дата обращения: 20.05.2026).
6. SQLite Consortium. SQLite Documentation [Электронный ресурс] // SQLite. — URL: https://www.sqlite.org/docs.html (дата обращения: 20.05.2026).
7. aiogram. FSM (Finite State Machine) [Электронный ресурс] // aiogram documentation. — URL: https://docs.aiogram.dev/en/v3.28.2/dispatcher/fsm/index.html (дата обращения: 20.05.2026).
8. Кубанский государственный университет. Приемная кампания [Электронный ресурс] // Официальный сайт КубГУ. — URL: https://www.kubsu.ru (дата обращения: 25.05.2026).

List of literature:

1. Telegram. Bot API Reference [Electronic resource] // Official Telegram documentation. — URL: https://core.telegram.org/bots/api (date of access: 05/20/2026).
2. aiogram. Official Documentation [Electronic resource] // aiogram framework. — URL: https://docs.aiogram.dev/en/v3.28.2 / (date of access: 05/20/2026).

3. Telegram. BotFather [Electronic resource] // Official Telegram documentation. — URL: https://core.telegram.org/bots#botfather (date of request: 05/20/2026).
4. Federal Law No. 152-FZ of 27.07.2006 "On Personal Data" [Electronic resource] // ConsultantPlus. — URL: https://www.consultant.ru/document/cons_doc_LAW_61801 / (accessed: 05/22/2026).
5. Python Software Foundation. asyncio — Asynchronous I/O [Electronic resource] // Python Documentation. — URL: https://docs.python.org/3/library/asyncio.html (accessed: 05/20/2026).
6. SQLite Consortium. SQLite Documentation [Electronic resource] // SQLite. — URL: https://www.sqlite.org/docs.html (date of request: 05/20/2026).
7. aiogram. FSM (Finite State Machine) [Electronic resource] // aiogram documentation. — URL: https://docs.aiogram.dev/en/v3.28.2/dispatcher/fsm/index.html (date of application: 05/20/2026).
8. Kuban State University. Admission campaign [Electronic resource] // Official website of KubGU. — URL: https://www.kubsu.ru (date of request: 05/25/2026).