Обзорная статья
УДК 689
EDN WSYJBK
https://doi.org/10.34216/2587-6147-2024-2-64-59-63
Даниил Вячеславович Скачков[1]
Юрий Леонидович Лустгартен[2]
[1,2]Костромской государственный университет, г. Кострома, Россия
[1]dan-s.skachkov@yandex.ru, https://orcid.org/0009-0000-5196-4588
[2]ylust@yandex.ru, https://orcid.org/0000-0002-3200-4490

АРХИТЕКТУРА ПРИЛОЖЕНИЯ КОНСТРУКТОРА БОТОВ ДЛЯ БИЗНЕС-ПРОЦЕССОВ

Аннотация. *В статье рассматривается проблема построения архитектуры приложения для эффективной интеграции ботов в бизнес-процессы. Авторы проводят анализ возможностей разработки ботов с использованием популярного мессенджера Telegram. Оцениваются аналоги трех основных конструкторов ботов (BotMan, PuzzleBot и SAMBOT), что позволяет определить их преимущества и недостатки. Рассматриваются различные архитектурные подходы, включая N-слойную, гексагональную, луковую и чистую, с целью выбора наиболее подходящего для данной задачи. Наконец, предлагается структурированное решение для back-end приложения на .NET с использованием ASP.NET. Предложенный подход обеспечивает эффективную интеграцию ботов в бизнес-процессы благодаря тщательно продуманной архитектуре.*
Ключевые слова: *разработка конструктора, возможности конструкторов, архитектура конструктора Telegram ботов, архитектура, луковая архитектура, монолитная архитектура, бизнес-процесс*

Для цитирования: Скачков В. Д., Лустгартен Ю. Л. Архитектура приложения конструктора ботов для бизнес-процессов // Технологии и качество. 2024. № 2(64). С. 59–63. https://doi.org/10.34216/2587-6147-2024-2-64-59-63.

Review article
Daniil V. Skachkov[1]
Yuriy L. Lustgarten[2]
[1,2]Kostroma State University, Kostroma, Russia

ARCHITECTURE OF THE BOT DESIGNER APPLICATION FOR BUSINESS PROCESSES

Abstract. *The article discusses the problem of building an application architecture for the effective integration of bots into business processes. The authors analyze the possibilities of developing bots using the popular Telegram messenger. The analogues of the three main bot constructors (Batman, PuzzleBot and SAMBOT) are evaluated, which allows us to determine their advantages and disadvantages. Various architectural approaches, including N-layer, hexagonal, onion and pure, are considered in order to choose the most suitable for this task. Finally, a structured solution is proposed for the back-end of the application on .NET using ASP.NET . The proposed approach ensures effective integration of bots into business processes thanks to a carefully thought-out architecture.*
Keywords: *designer development, designer capabilities, Telegram bot designer architecture, architecture, onion architecture, monolithic architecture, business process*

For citation: Skachkov D. V., Lustgarten Yu. L. Architecture of the bot designer application for business processes. Technologies & Quality. 2024. No 2(64). P. 59–63. (In Russ.). https://doi.org/10.34216/2587-6147-2024-2-64-59-63.

Боты стали неотъемлемой частью общения в современном цифровом мире. Эти автоматизированные программы предлагают широкий спектр услуг от предоставления информации до выполнения услуг и развлечения пользователей. Одними из популярных являются Телеграм-боты. Для их создания используются различные платформы и инструменты. Разработчики могут использовать следующие методы:

– официальный Telegram Bot API [1] для прямого взаимодействия с платформой Telegram через HTTPS-протокол;

– сторонние библиотеки и фреймворки, которые упрощают процесс разработки;
– конструкторы ботов, для которых не требуются навыки программирования.

Разрабатываемая система является аналогом конструктора. Одно из требований к системе – пользователь без опыта программирования должен иметь возможность создавать и настраивать ботов в соответствии с конкретными требованиями своего бизнеса. Для учета в разрабатываемой системе преимуществ и недостатков существующих конструкторов был проведен их анализ. Среди основных аналогов [2] можно выделить BotMan [3], PuzzleBot [4] и SAMBOT [5] с точки зрения наилучшего функционала и аудитории. Анализ и тестирование существующих конструкторов чат-ботов выявил их неспособность удовлетворять современным потребностям, таким как работа с собственной базой данных, интеграция с информационными источниками, поддержка различных платформ обмена сообщениями и гибкая настройка ботов.

Ограничения существующих конструкторов в удовлетворении этих потребностей приводят к зависимости от сторонних сервисов, сложностям в интеграции и недостаточной гибкости. Это подчеркивает необходимость в новой системе конструкторов, которая преодолевает эти ограничения и предоставляет пользователям расширенные возможности для создания ботов, отвечающих их уникальным требованиям.

Для создания масштабируемых, гибких и надежных ботов, которые могут интегрироваться с различными системами и адаптироваться к изменяющимся требованиям, необходим более комплексный подход к проектированию архитектуры. Общая архитектура всей системы представлена на рис. 1. Система состоит из нижеперечисленных основных компонентов.

1. ClientApplication – клиентское desktop или веб-приложение. Представляет пользовательский интерфейс системы. Принимает и передает данные от API системы через протокол HTTPS.

2. LocalMessangerAPI – back-end приложение на ASP.NET. Обрабатывает входящие запросы, выполняет задачи, взаимодействует с базой данных для хранения и извлечения данных и интегрируется с внешними сервисами для расширения функциональности ботов. Подходом, выбранным для разработки, является REST (Representational State Transfer). REST был выбран в связи с тем, что нет требования по скорости выполнения запросов и основной упор делается на гибкость и простоту реализации.

3. Postgres SQL Server – сервер, на котором расположена база данных. Выбранной СУБД является Postgres SQL. Она более масштабируемая, чем другие, предоставляет большой набор функций. На данном сервере будут находиться две базы данных. Первая для LocalMessangerAPI, вторая для TelegramAPI.

4. TelegramAPI – сервис-прослойка для обмена сообщениями между Telegram и конструктором по протоколу HTTPS. При отправке сообщений в Telegram данный сервис получает обновления и отправляет их на LocalMessangerAPI.

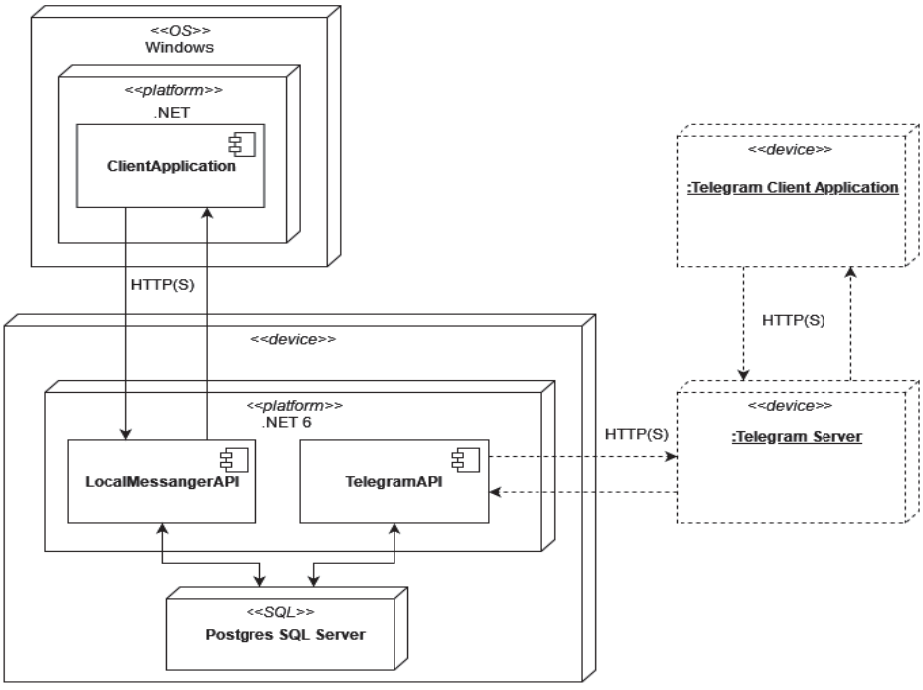

Рис. 1. Архитектура системы

Сама по себе система будет иметь модульную монолитную архитектуру. Микросервисная архитектура требует управления несколькими кластерами, каждый из которых запускает один или несколько микросервисов. Это может увеличить сложность и накладные расходы на управление. Кроме этого, монолитную модульную архитектуру проще разрабатывать, поскольку все компоненты объединены в один физический модуль, а внутри системы деление идет на несколько модулей.

Существует множество архитектур решения для различных систем. Среди основных известных можно выделить такие архитектуры, как N-слойная, луковая, гексагональная, чистая [6, 7]. Для выбора архитектуры был проведен сравнительный анализ, который представлен в таблице.

Таблица

Сравнительный анализ архитектур решений

Характеристика	N-слойная	Луковая	Гексагональная	Чистая
Структура	Плоские слои с зависимостями между ними	Концентрические слои	Порты и адаптеры	Центр, внутренний и внешний круги
Фокус	Разделение обязанностей и тестируемость	Изоляция доменной логики	Гибкость и независимость	Независимость и модульность
Связь между слоями	Связи между слоями	Внутренние слои зависят от внешних	Адаптеры связывают порты и приложения	Внешний круг зависит от внутреннего
Тестируемость	Средняя	Высокая	Высокая	Высокая
Модульность	Средняя	Средняя	Высокая	Высокая
Гибкость	Средняя	Средняя	Высокая	Низкая
Сложность реализации	Низкая	Низкая	Высокая	Средняя
Интуитивность	Высокая	Высокая	Низкая	Средняя

Анализ каждой из архитектур показал, что N-слойная, гексагональная и чистая архитектуры избыточны для проекта конструктора ботов, так как представляют собой более сложные модели, чем луковая, в следующих аспектах:

– сложность и трудоемкость реализации: гексагональная и чистая архитектуры требуют больших затрат времени и усилий для реализации, что не соответствует простоте и скорости разработки, необходимым для конструктора ботов;
– потребность в более детальном изучении. Применение гексагональной и чистой архитектур требует более глубокого понимания их тонкостей и специфики реализации. Это может отнимать дополнительное время на изучение и проектирование, что нецелесообразно в данном проекте;
– ограниченная гибкость: N-слойная архитектура может ограничивать гибкость в интеграции с внешними сервисами, что важно для конструктора ботов, который может использовать различные API и интеграции.

С учетом новых потребностей системы была выбрана луковая архитектура по следующим причинам:

1) позволяет легко работать с базам данных различной структуры благодаря четкому разделению между доменом и инфраструктурой;

2) упрощает интеграцию с различными информационными системами за счет слабосвязанных компонентов и использования абстракций и адаптеров;

3) позволяет легко подключать систему конструкторов к различным внешним системам и сервисам, которые могут предоставлять данные, функциональность или интеграцию с другими платформами, сохраняя целостность и независимость доменного слоя;

4) обеспечивает гибкость в настройке ботов под конкретные чаты или группы пользователей, поскольку доменный слой не зависит от внешних факторов и легко адаптируется к меняющимся требованиям.

Таким образом, луковая архитектура предоставляет прочную и масштабируемую основу для системы конструкторов, удовлетворяя ее потребности в гибкости, интеграции и поддержке различных платформ и источников данных. Структура решения представлена на рис. 2. Строгие зависимости не были применены по причине того, что это увеличивает сложность реализации системы. Зависимости проектов извне вовнутрь:

1) Domain не имеет зависимостей;

2) Application зависит от Domain;

3) Infrastructure зависит от Domain и Application;

4) WebApi зависит от Application и Infrastructure.

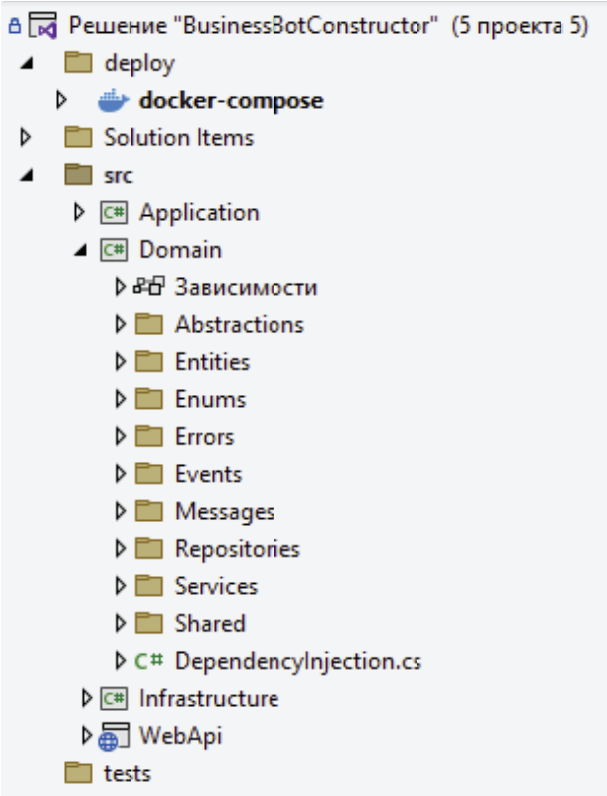

Рис. 2. Структура решения в Visual Studio

Были изучены и сравнены различные архитектурные подходы, включая N-слойную архитектуру, гексагональную архитектуру, чистую архитектуру и луковую архитектуру. После оценки применимости каждого подхода к требованиям телеграм-бота была выбрана луковая архитектура из-за ее преимуществ в обеспечении слабой связи между компонентами, устойчивости к изменениям и возможности горизонтального масштабирования. Затем реализована луковая архитектура для системы конструкторов бота, включая определение слоев, компонентов и их взаимодействий. Далее разработаны основные доменные сущности для будущего использования в системе.

ВЫВОДЫ

Анализ различных конструкторов ботов (BotMan, PuzzleBot и SAMBOT) позволяет разработчикам выбрать решение, которое наилучшим образом соответствует их конкретным требованиям. Но этого недостаточно для современных потребностей, таких как интеграция с внешними информационными системами и подключение собственных баз данных. Четко определенная архитектура является основой для создания конструктора ботов. Архитектура определяет структуру и взаимодействие компонентов бота, обеспечивая его производительность, масштабируемость, безопасность и простоту обслуживания. Луковая архитектура была выбрана для данного телеграм-бота из-за ее преимуществ в обеспечении слабой связи между компонентами, устойчивости к изменениям и возможности горизонтального масштабирования. Предложенный в статье подход к построению архитектуры приложения в будущем обеспечит эффективную интеграцию ботов в бизнес-процессы благодаря тщательно продуманному выбору конструкторов ботов, архитектурных подходов и реализации на базе .NET с использованием ASP.NET и луковой архитектуры решения.

СПИСОК ИСТОЧНИКОВ

1. Telegram Core : сайт. URL: https://core.telegram.org (дата обращения: 05.02.2024).
2. Какой конструктор чат-ботов выбрать? Сравнение 67 сервисов для Telegram // ВКонтакте : соц. сеть. URL: https://vc.ru/services/389239-kakoy-konstruktor-chat-botov-vybrat-sravnenie-67-servisov-dlya-telegram (дата обращения: 16.02.2024).
3. BotMan. URL: https://botman.pro (дата обращения: 17.02.2024).
4. PuzzleBot. URL: https://puzzlebot.top (дата обращения: 17.02.2024).
5. SAMBOT. URL: https://sambot.ru (дата обращения: 17.02.2024).
6. Hexagonal vs Clean vs Onion Architectures: Choosing the Right Architecture. URL: https://programmingpulse.vercel.app/blog/hexagonal-vs-clean-vs-onion-architectures (дата обращения: 21.02.2024).
7. Эволюция серверной архитектуры: N-слойная, DDD, шестиугольная, луковичная, чистая // NOP: Nuances of programming. Образовательные статьи и переводы – все для программиста. URL: https://nuancesprog.ru/p/18856 (дата обращения: 21.02.2024).

REFERENCES

1. Telegram Core. URL: https://core.telegram.org (accessed 05.02.2024).
2. Which chatbot designer should I choose? Comparison of 67 services for Telegram. URL: https://vc.ru/services/389239-kakoy-konstruktor-chat-botov-vybrat-sravnenie-67-servisov-dlya-telegram (accessed 16.02.2024).

3. Batman. URL: https://botman.pro (accessed 17.02.2024).
4. PuzzleBot. URL: https://puzzlebot.top (accessed 17.02.2024).
5. SAMBOT. URL: https://sambot.ru (accessed 17.02.2024).
6. Hexagonal vs Clean vs Onion Architecture: Choosing the Right Architecture. URL: https:// programmingpulse.vercel.app/blog/hexagonal-vs-clean-vs-onion-architectures (accessed 21.02.2024).
7. The evolution of the server architecture: n-layer, DDD, hexagonal, bulbous, pure. URL: https://nuancesprog.ru/p/18856 (accessed 21.02.2024).

Статья поступила в редакцию 19.05.2024
Принята к публикации 24.05.2024