АННОТАЦИЯ

Расчетно-пояснительная записка 73 с., 39 рис., 10 табл., 14 источников.

LOW-CODE ПЛАТФОРМА, API TELEGRAM, MVVM, ЧАТ-БОТ, КОНСТРУИРОВАНИЕ ЧАТ-БОТОВ, КОНФИГУРАЦИЯ, АВТОМАТИЗАЦИЯ БИЗНЕС – ПРОЦЕССОВ, ТЕСТ-КЕЙСЫ, ТЕСТИРОВАНИЕ

Объектом разработки является low-code платформа для разработки чат-ботов в мессенджере Telegram.

Цель работы – предоставить возможность пользователям без опыта программирования создавать чат-ботов, а также помочь компаниям снизить зависимость от разработчиков и ускорить процесс создания программного обеспечения.

Цель работы достигается за счет разработки интуитивно понятного конструктора, который позволяет пользователям без опыта программирования создавать чат-ботов. Вместо написания кода, пользователь может использовать графический интерфейс и функциональные блоки для создания и настройки функциональности ботов. Также цель работы достигается за счет интеграции с API Telegram, что позволяет ботам взаимодействовать с популярной мессенджер-платформой.

Все эти меры помогают компаниям снизить зависимость от разработчиков и ускорить процесс создания программного обеспечения, так как создание приложений становится доступным для широкого круга пользователей, без необходимости в специализированных навыках написания сценариев для чат-ботов с использованием языков программирования.

**СОДЕРЖАНИЕ**

[АННОТАЦИЯ 2](#_Toc137796103)

[ВВЕДЕНИЕ 4](#_Toc137796104)

[1. НАУЧНО – ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ 5](#_Toc137796105)

[1.1 Постановка задачи проектирования 5](#_Toc137796106)

[1.2 Описание предметной области 6](#_Toc137796107)

[1.3 Анализ аналогов и прототипов 8](#_Toc137796108)

[1.4 Перечень задач, подлежащих решению в процессе разработки 13](#_Toc137796109)

[1.5 Обоснование выбора инструментов и платформы для разработки 13](#_Toc137796110)

[1.6 Техническое задание 19](#_Toc137796111)

[2. ПРОЕКТНО – КОНСТРУКТОРСКАЯ ЧАСТЬ 27](#_Toc137796112)

[2.1 Разработка структуры приложения 27](#_Toc137796113)

[2.2 Диаграмма классов 30](#_Toc137796114)

[2.3 Разработка алгоритмов обработки информации 31](#_Toc137796115)

[2.4 Разработка интерфейса взаимодействия пользователя с системой 41](#_Toc137796116)

[3. ПРОЕКТНО – ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ 46](#_Toc137796117)

[3.1 Тестирование и отладка рабочей программы 46](#_Toc137796118)

[3.2 Порядок развертывания системы 50](#_Toc137796119)

[3.3 Руководство пользователя 52](#_Toc137796120)

[3.4 Экономическая часть 64](#_Toc137796121)

[3.5 Эргономика проектного решения 68](#_Toc137796122)

[ЗАКЛЮЧЕНИЕ 70](#_Toc137796123)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 71](#_Toc137796124)

[ПРИЛОЖЕНИЕ А 73](#_Toc137796125)

# ВВЕДЕНИЕ

В современном цифровом мире мессенджеры играют важную роль в нашей повседневной коммуникации. Они предоставляют удобный и мгновенный способ общения, позволяют делиться информацией, получать уведомления и проводить бизнес-операции. Один из наиболее популярных мессенджеров, Telegram, обладает богатым функционалом и широкими возможностями для разработчиков.

Современные технологические тренды и стремление к упрощению процессов разработки программного обеспечения привели к росту интереса к концепции "low-code". Подход "low-code" описывает методику создания приложений и систем с минимальным уровнем программирования, где основная работа выполняется платформой, предоставляющей готовые модули и инструменты.

В связи с растущим спросом на создание чат-ботов в мессенджерах возникает потребность в разработке инструментов, позволяющих широкому кругу пользователей, включая тех, кто не обладает программированием, создавать свои собственные приложения. В этом контексте low-code платформы становятся все более популярными, предлагая удобные визуальные инструменты и минимальное количество кода для разработки приложений.

Целью данной дипломной работы является разработка low-code платформы для создания ботов в мессенджере Telegram. Это приложение позволит пользователям без опыта программирования легко и быстро создавать разнообразные боты с поддержкой различных команд и функций. Платформа имеет важное значение в контексте развития бизнеса, так как она позволяет компаниям снизить зависимость от разработчиков и значительно ускорить процесс создания программного обеспечения.

1. НАУЧНО – ИССЛЕДОВАТЕЛЬСКАЯ ЧАСТЬ

1.1 Постановка задачи проектирования

Задачей является разработка low-code платформы для создания ботов в мессенджере Telegram, которая позволяет пользователям без опыта программирования легко и быстро создавать различных чат-ботов с любым поддерживаемым количеством команд. В рамках проекта требуется реализовать следующие модули и функциональности:

1. Пользовательский интерфейс: Разработка интуитивно понятного и удобного пользовательского интерфейса, который позволяет пользователям визуально создавать и настраивать ботов. Интерфейс должен предоставлять инструменты для добавления команд, кнопок, обработки входящих сообщений и других элементов функциональности бота.
2. Модуль для создания бота: Разработка модуля, который позволяет пользователям определить логику работы бота, задать его поведение и настройки. Этот модуль должен поддерживать создание различных типов команд, обработку входящих сообщений, управление состоянием бота и другие функциональные возможности.
3. Модуль отладки: Разработка инструментов и функциональности для отладки созданных ботов. Этот модуль должен предоставлять возможность проверить и протестировать логику работы бота, обнаружить и исправить ошибки, а также предоставлять детальную информацию о процессе работы бота для анализа и улучшения его функциональности.
4. Модуль взаимодействия с данными: данный модуль должен обеспечивает взаимодействие с внешними источниками данных, API и другими системами. Этот модуль должен предоставлять функциональность для сохранения и получения данных, обновления состояния бота, а также обеспечивать безопасность и надежность обработки данных.

Целью разработки платформы является предоставление простого, гибкого и мощного инструмента для создания ботов в мессенджере Telegram, который позволит пользователям без программирования реализовывать свои идеи и потребности. Реализация указанных модулей и функциональности будет обеспечивать эффективное создание и настройку ботов, упрощение процесса разработки и увеличение гибкости и масштабируемости приложений.

1.2 Описание предметной области

Конструктор чат-ботов представляет собой мощный инструмент, предназначенный для создания виртуальных помощников. Этот инструмент позволяет пользователям самостоятельно настраивать ботов, даже без предварительного знания языков программирования. Таким образом, он становится доступным каждому человеку, не зависимо от его технического опыта.

Одной из главных целей использования конструктора чат-ботов в бизнесе является автоматизация процесса коммуникации с клиентами. Благодаря заранее продуманным сценариям, виртуальные помощники могут помочь бизнесу отправлять новости, напоминания или обновления, обрабатывать заказы, предоставлять поддержку клиентам и выполнять множество других функций.

Основная функция чат-ботов заключается в предоставлении быстрых и четких ответов на определенные вопросы, для которых боты имеют необходимую информацию. Таким образом, пользователи могут получить нужные ответы и решения в кратчайшие сроки.

Для реализации бота понадобиться:

1. Low-code платформа: платформа должна предоставлять визуальный конструктор и инструменты для создания ботов без необходимости в глубоких знаниях программирования.
2. Доступ к Telegram Bot API [2]: понадобится доступ к Telegram Bot API, который позволяет взаимодействовать с Telegram и управлять ботом.
3. Конструктор ботов: воспользоваться конструктором, предоставляемым low-code платформой, чтобы настроить логику и функциональность чат-бота. Этот конструктор позволяет создавать различные действия, ответы на сообщения пользователей, управлять диалогами и многое другое.
4. Тестирование и развертывание: после создания бота с помощью low-code платформы, провести тестирование его функциональности и корректности работы. Затем развернуть бота на платформе Telegram, чтобы он стал доступен для пользователей.

Основные преимущества использования чат-ботов:

* Обработка типовых запросов: Чат-боты являются отличным инструментом для обработки большого количества типовых запросов от пользователей. Они могут предоставлять стандартные ответы на часто задаваемые вопросы, что существенно ускоряет обработку запросов и снижает нагрузку на операторов.
* Непрерывная доступность: Чат-боты работают круглосуточно, 24 часа в сутки, без необходимости ориентироваться на часы работы операторов. Пользователи могут задать вопросы и получить ответы в любое время суток, что значительно повышает удобство и доступность обслуживания.
* Сбор данных и аналитика: Чат-боты позволяют эффективно собирать данные о пользователях и их потребностях. Вся статистика по обращениям фиксируется автоматически, если настроены соответствующие параметры. Это позволяет анализировать поведение пользователей, выявлять тренды и предоставлять более персонализированное обслуживание.
* Низкие затраты на обслуживание: Чат-боты требуют низких затрат на техническое обслуживание. Разработав и запустив чат-бота один раз, его можно относительно просто расширять и обновлять. Это существенно экономит затраты по сравнению с переобучением сотрудников колл-центра и обновлением скриптов ответов при изменениях в сценарии.
* Улучшение пользовательского опыта: Чат-боты позволяют достичь более высокого уровня обслуживания и удовлетворения пользователей. Быстрые и четкие ответы на запросы, доступность в любое время, персонализированный подход и возможность оперативной обратной связи делают пользовательский опыт более удобным и приятным.
* Эти преимущества делают чат-боты эффективным инструментом для автоматизации коммуникации с клиентами, улучшения процессов обслуживания и снижения операционных затрат.

После реализации системы создание виртуальных помощников упростится, потому что не надо будет изучать большое количество материала по созданию бота вручную, можно будет воспользоваться удобным и самое главное понятным для обычного человека интерфейсом. Платформа не только облегчит создание, но и сможет автоматизировать различные бизнес-процессы такие как поддержка клиентов, предоставление информации о товарах и оформление заявок.

1.3 Анализ аналогов и прототипов

**SendPulse** - это одна из популярных маркетинговых платформ, предоставляющая широкий спектр инструментов для взаимодействия с аудиторией в мессенджере Telegram. Среди множества аналогичных решений, SendPulse выделяется своими удобными и гибкими возможностями.

Одним из ключевых преимуществ SendPulse является его многофункциональность. Платформа предоставляет возможность отправлять как текстовые, так и мультимедийные сообщения, включая изображения, видео и аудиофайлы. Это позволяет создавать более привлекательные и интерактивные боты, обогащая пользовательский опыт.

Кроме того, SendPulse обладает интегрированным конструктором для создания ботов без необходимости программирования. Это означает, что даже люди без технических навыков могут легко настроить и настроить своего собственного чат-бота. Конструктор предлагает широкий выбор функций и настроек, таких как автоматические ответы, кнопки, вопросы и ответы, оповещения и многое другое, что делает создание бота быстрым и удобным процессом.

Одной из значимых особенностей SendPulse является его поддержка различных типов интеграций и API. Это позволяет разработчикам интегрировать бота SendPulse со своими существующими системами и приложениями, расширяя его функциональность и применение в различных сферах бизнеса.

Чтобы убрать логотип компании со всех виджетов и отправлять неограниченное количество сообщений, необходимо приобрести план “WHITE LABEL” за 599 рублей в месяц.

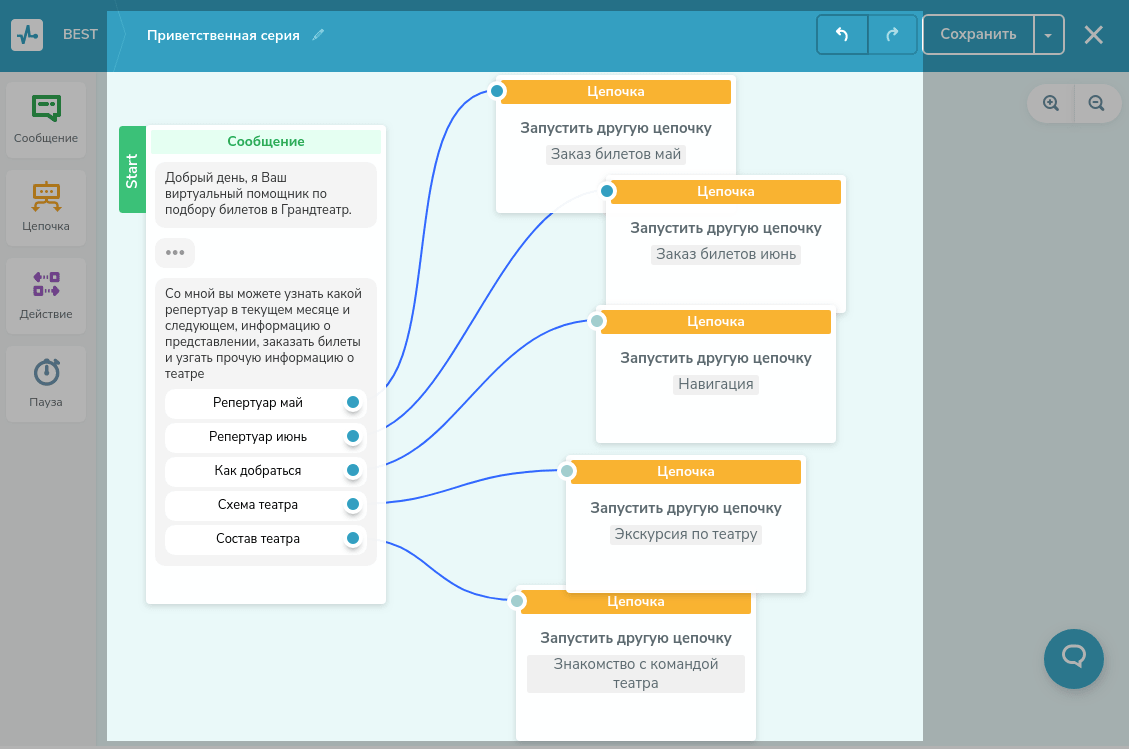


Рисунок 1.3.1 – Связи между сообщениями и различными действиями в платформе SendPulse.

**Aimylogic** предоставляет возможность создавать чат-ботов для мессенджера Telegram с помощью своей платформы. Они предлагают два плана: бесплатный план "Community" и платные тарифы с расширенным функционалом.

При использовании бесплатного плана "Community" вы можете создавать чат-ботов, но функционал этого плана ограничен. В рамках данного плана можно использовать до тридцати фраз, однако доступ к текстам переписки отсутствует, и нельзя подключить оператора к беседе. Также, дополнительные возможности, такие как расширенная статистика, не доступны в бесплатном плане.

Для расширения функционала и получения дополнительных возможностей рекомендуется перейти на один из платных тарифов. Стоимость пакетов начинается от 990 рублей в месяц. Платные тарифы предлагают больше функций и возможностей, включая более гибкую настройку чат-ботов, подключение операторов, доступ к полной статистике и другие преимущества.

Выбор платного тарифа зависит от ваших потребностей и ожидаемого функционала. Стоимость пакетов может варьироваться в зависимости от выбранного тарифного плана и дополнительных услуг, которые могут быть включены в пакет.

При использовании Aimylogic можно создавать чат-ботов в Telegram с учетом ваших потребностей и требований, получая возможность автоматизировать коммуникацию с пользователями и улучшить опыт взаимодействия с вашим брендом или бизнесом.

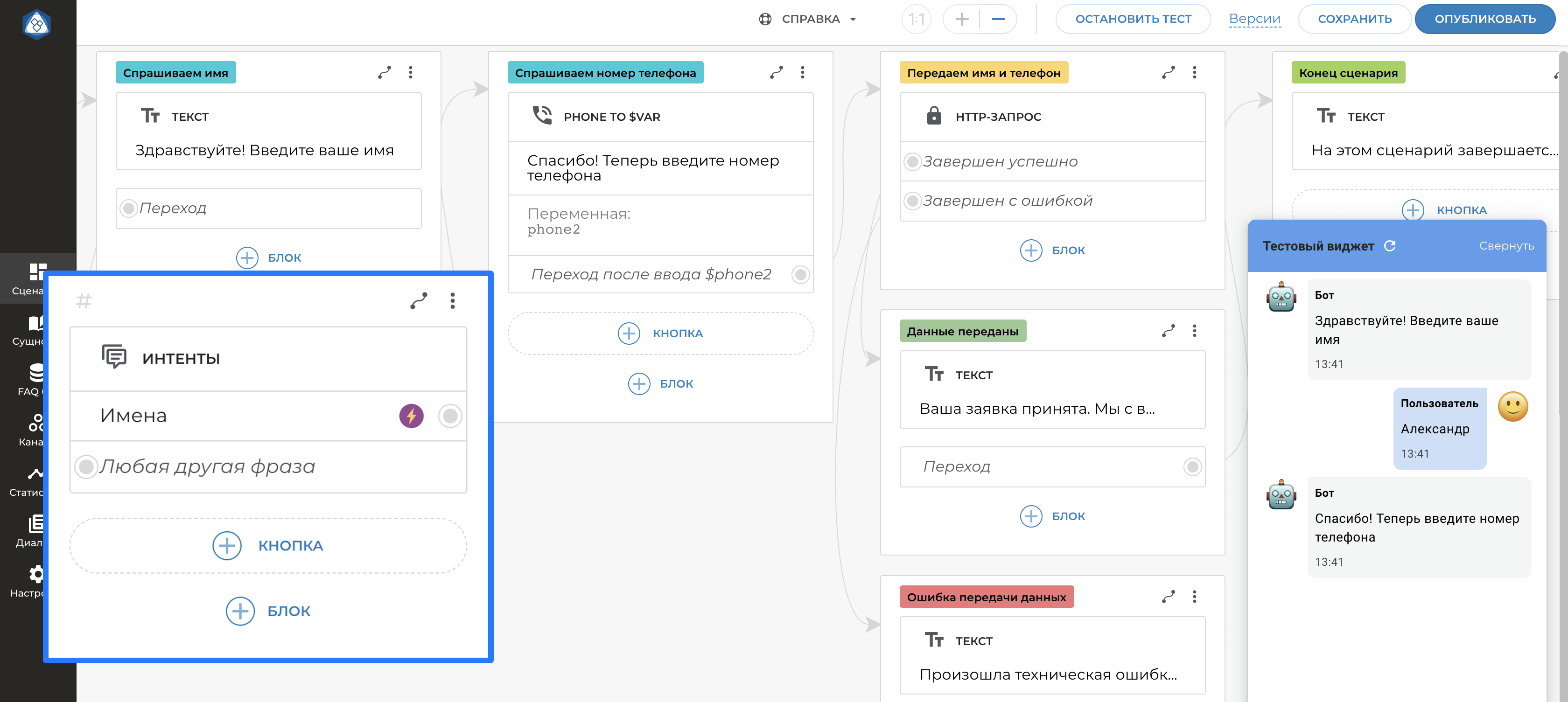


Рисунок 1.3.2 – процесс создания бота с использованием платформы Aimlogic.

**BotKits** предлагает бесплатный пробный период продолжительностью 7 дней, в течение которого пользователи могут полностью использовать функционал платформы без каких-либо ограничений. Для создания одного чат-бота требуется оплата в размере 499 рублей, а для создания двух чат-ботов — 949 рублей. Существует также пакет Pro, доступный за 8499 рублей в месяц, который предоставляет дополнительные возможности, включая отслеживание статистики и возможность подключения менеджера к диалогам.

Таблица 1.3.1 – Стоимость создание чат-бота по поручению клиента на платформе BotKits

|  |  |  |  |
| --- | --- | --- | --- |
| Обслуживание | | | |
| Абонентская плата. руб. в месяц | 1930₽/ 25$ | 3860₽/ 50$ | 7700₽/ 100$ |
| Минимальная оплата за 3 месяца | 5800₽/ 75$ | 11600₽/ 150$ | 23100₽/ 300$ |
| Внесение правок после реализованного ТЗ | Не более 1 правки | Не более 3 правок | Не более 5 правок |

**Botmother** предлагает широкий выбор тарифных планов, среди которых четыре основных варианта. Первый план является бесплатным и предоставляет возможность создать одного бота. Однако, в рамках данного плана ограничен доступ к статистике, невозможно добавить менеджера в беседу и использовать функцию рассылок.

Для пользователей, желающих получить больше функциональности и возможностей, Botmother предлагает платные тарифные пакеты. Стоимость этих пакетов начинается от 1499 рублей в месяц. Платные тарифы позволяют настроить не только одного бота, но и предоставляют расширенный доступ к статистике, возможность добавления менеджера в беседу и использование функции рассылок.

Таким образом, пользователи Botmother могут выбрать подходящий тарифный план в зависимости от своих потребностей и бюджета, получая при этом доступ к различным функциям и инструментам для создания и управления своими ботами в мессенджере Telegram.

**13chats** представляет собой удобный онлайн-сервис, созданный для маркетологов и команд поддержки, с целью увеличения продаж через использование онлайн-чата, умных виджетов и чат-ботов. С помощью данного сервиса можно повысить эффективность взаимодействия с клиентами и улучшить процессы продаж.

Одним из преимуществ 13chats является доступность бесплатного тарифа, который позволяет использовать сервис с полным функционалом для до 5 операторов. Это отличная возможность ознакомиться с возможностями платформы и оценить ее применимость для ваших целей.

Кроме того, сервис предлагает различные тарифные планы, которые обеспечивают расширенные функциональные возможности и больший объем операторов. Стандартный тариф позволяет работать с 13 операторами и доступен по цене $13 в месяц. Тариф "Про" предоставляет еще больше возможностей и рассчитан на команды с более высокими требованиями, его стоимость составляет $33 в месяц. Для пользователей, нуждающихся в дополнительных преимуществах и возможностях, доступен тариф "Премиум" за $93 в месяц.

Важно отметить, что при необходимости вы можете запросить увеличение лимитов в рамках выбранного тарифного плана, чтобы адаптировать сервис под ваши специфические потребности и требования.

С использованием 13chats вы получите мощный инструмент для улучшения коммуникации с клиентами, увеличения продаж и повышения уровня обслуживания. Он обеспечивает удобство, эффективность и гибкость взаимодействия с вашей целевой аудиторией, помогая достичь поставленных бизнес-целей.

1.4 Перечень задач, подлежащих решению в процессе разработки

В процессе разработки системы должны быть реализованы следующие задачи:

1. Создание, редактирование проектов.
2. Логирование проекта.
3. Добавление, редактирование удаление команд.
4. Набор настроек по типу команд.
5. Возможность запустить текущий проект для просмотра функционала сконструированного бота
6. Формирование итогового проекта в удобно запускаемую программу.
7. Мультиязычная система (с возможностью расширения при необходимости).
8. Тестирование и написание тест кейсов.

1.5 Обоснование выбора инструментов и платформы для разработки

**Интерфейс**

В связи с тем, что целевая аудитория программы имеет базовые компьютерные навыки, необходимо создать интерфейс, который будет интуитивно понятным и удобным в использовании. Для этой цели была выбрана технология WPF (Windows Presentation Foundation)[10].

WPF предоставляет несколько преимуществ, делающих его подходящим выбором. Кроме того, WPF имеет веб-подобную модель компоновки, позволяющую гибко располагать элементы управления на основе их содержимого. Это позволяет создавать интерфейс, способный отображать высокодинамичное содержимое.

Еще одним преимуществом WPF является его декларативный пользовательский интерфейс. Каждое окно программы описывается в виде XML-документа с использованием языка разметки XAML. Это позволяет полностью отделить пользовательский интерфейс от кода программы, что упрощает его разработку, сопровождение и модификацию.

**Языки программирования**

Для реализации данного приложения были выбраны два языка программирования: Python и C#. Каждый из них имеет свои преимущества и обладает специфическими возможностями.

Одним из главных преимуществ языка Python является его высокая скорость разработки. Благодаря простому и понятному синтаксису Python, разработчикам требуется меньше времени на написание кода. Кроме того, в языке Python существует множество фреймворков и библиотек, которые упрощают разработку и расширение приложения.

Язык программирования C# также имеет свои преимущества, особенно для реализации данного приложения. Одно из ключевых преимуществ C# заключается в его поддержке WPF (Windows Presentation Foundation), что позволяет создавать удобные и эффективные пользовательские интерфейсы. Кроме того, благодаря работе на машине CLR (Common Language Runtime), приложение, разработанное на C#, запускается на виртуальном процессоре. Это означает, что ошибки, возникающие в приложении, не влияют на работу других программ в системе. Однако, для запуска программы на C# требуется некоторое дополнительное время.

Таким образом, использование языков программирования Python и C# позволяет создать надежное и эффективное приложение. Python обеспечивает быструю разработку и обширные возможности библиотек, а C# обеспечивает поддержку WPF и надежность выполнения программы.

**Среда разработки**

И средой для разработки была выбрана Microsoft Visual Studio 2022. Ее набор инструментов разработки, основанных на использовании компонентов, и других технологий позволяет создавать мощные, производительные приложения. Кроме того, среда Visual Studio оптимизирована для совместного проектирования, разработки и развертывания корпоративных решений. Visual Studio предоставляет средства для проектирования, разработки и отладки. Немало важным плюсом является поддержка, рефакторинг используемых языков таких как C# и Python

А также такие достоинства как:

1. Visual Studio IntelliCode повышает эффективность разработки программного обеспечения с помощью искусственного интеллекта (ИИ). Для создания рекомендаций IntelliCode анализирует 2000 проектов с открытым кодом на GitHub
2. Рефакторинг В C#. Есть много новых удобных возможностей рефакторинга, которые помогают упорядочить код. Они отображаются как предложения со значком лампочки и включают такие действия, как перемещение элементов в интерфейс или базовый класс, настройку пространств имен в соответствии со структурой папок.

**Библиотека для логирования NLog**

При выборе библиотеки для регистрации и управления журналом событий (логирования) в проекте, было принято решение использовать библиотеку NLog. Этот выбор основан на следующих факторах:

Гибкость и настраиваемость NLog предоставляет широкие возможности для настройки журнала событий в соответствии с требованиями проекта. Он позволяет определять различные цели (targets) для записи логов, такие как файлы, базы данных, почта и другие. Кроме того, NLog поддерживает различные форматы логирования, что позволяет адаптировать журнал к нуждам проекта.

Высокая производительность NLog является эффективной библиотекой с минимальным влиянием на производительность приложения. Он обладает оптимизированным механизмом записи логов, что позволяет эффективно обрабатывать большие объемы информации при минимальной задержке.

NLog обладает обширной документацией и активной поддержкой, включающей примеры использования и руководства по настройке [5]. Кроме того, у NLog активное сообщество пользователей, что обеспечивает доступ к информации, помощь и обновления.

Большой функционал NLog предлагает широкий набор функций, включая фильтрацию, форматирование, поддержку контекста и другие возможности, которые облегчают процесс регистрации и анализа логов.

Исходя из вышеперечисленных причин, библиотека NLog была выбрана в качестве инструмента для регистрации и управления логами в проекте. Ее гибкость, производительность, поддержка различных платформ, документация и функционал обеспечивают надежное и эффективное логирование, что является важным компонентом успешного проекта.

**Библиотека aiogram**

При выборе библиотеки для взаимодействия с Telegram API в проекте было принято решение использовать aiogram. Этот выбор основан на следующих факторах:

Простота использования: aiogram предоставляет простой и интуитивно понятный интерфейс для работы с Telegram API. Она обладает четкой документацией и множеством примеров, что позволяет быстро освоить основные функции и начать разработку ботов [4].

Обширный функционал: aiogram предлагает широкий набор функций, необходимых для разработки ботов в Telegram. Она поддерживает работу с сообщениями, клавиатурами, обработку команд, встроенные типы данных и другие возможности, что облегчает создание разнообразного функционала в боте.

Гибкость и настраиваемость aiogram позволяет настраивать поведение и функциональность бота в соответствии с требованиями проекта. Она поддерживает обработку различных типов сообщений, возможность работы с медиа-файлами, настройку обработчиков событий и другие опции, что позволяет адаптировать бота под конкретные нужды проекта.

Активное сообщество и поддержка: aiogram имеет активное сообщество пользователей, что обеспечивает доступ к обновлениям, документации, руководствам и помощи. Это позволяет быстро решать возникающие проблемы и получать поддержку при разработке бота.

Производительность: aiogram разработана с учетом оптимизации и эффективности работы с Telegram API. Она предлагает асинхронный подход к обработке запросов, что позволяет обеспечить быструю и отзывчивую работу бота даже при высоких нагрузках.

Библиотека aiogram была выбрана в качестве инструмента для взаимодействия с Telegram API в проекте. Ее простота использования, обширный функционал, гибкость, поддержка сообщества и высокая производительность обеспечивают надежное и эффективное взаимодействие с платформой Telegram, что является важным компонентом успешного разработки ботов.

**Библиотека для тестирования XUnit**

При выборе библиотеки для тестирования в проекте было принято решение использовать Xunit. Этот выбор основан на следующих факторах:

Простота использования Xunit предоставляет простой и понятный синтаксис для написания тестов. Она обладает интуитивным подходом к организации тестовых классов и методов, что позволяет быстро создавать и поддерживать тестовый набор.

Обширный функционал Xunit предлагает широкий набор функций для тестирования, включая поддержку аннотаций, параметризованных тестов, тестовых атрибутов и многого другого. Она позволяет проверять различные аспекты кода, включая результаты, исключения, состояния и поведение[6].

Интеграция с Visual Studio и другими инструментами: Xunit интегрируется хорошо с различными средами разработки, включая Visual Studio, что облегчает запуск и отладку тестов. Она также поддерживает интеграцию с другими инструментами разработки, такими как CI/CD системы и средства непрерывной интеграции.

Гибкость и расширяемость Xunit предоставляет возможность создавать собственные расширения и настраивать поведение тестового фреймворка под нужды проекта. Она поддерживает использование сторонних библиотек и инструментов, что позволяет расширить функциональность и улучшить процесс тестирования.

Активное сообщество и поддержка Xunit имеет активное сообщество пользователей, которое обеспечивает доступ к обновлениям, документации, руководствам и поддержке. Это позволяет получить помощь и решить возникающие проблемы при разработке и поддержке тестов.

Исходя из вышеперечисленных причин, библиотека Xunit была выбрана в качестве инструмента для тестирования в проекте. Ее простота использования, обширный функционал, интеграция с инструментами разработки, гибкость и поддержка сообщества обеспечивают эффективное и надежное тестирование кода, что является важным компонентом успешной разработки программного обеспечения.

**Библиотека Material Design**

Выбор Material Design для WPF обоснован несколькими факторами [1]. Во-первых, эта библиотека предоставляет широкий набор готовых элементов дизайна, таких как кнопки, текстовые поля, списки и другие, которые можно легко интегрировать в приложение. Они имеют современный и стильный внешний вид, что способствует созданию привлекательного интерфейса.

Во-вторых, Material Design для WPF обладает хорошей документацией и поддержкой сообщества. Это позволяет разработчикам быстро освоить библиотеку, получить необходимую информацию и решить возникающие проблемы.

Кроме того, использование Material Design в проекте также обеспечивает единообразный и согласованный стиль интерфейса, что важно для пользователей. Они будут ощущать привычность и комфорт при использовании приложения, так как Material Design широко распространен и знаком им.

Таким образом, выбор библиотеки Material Design для WPF обоснован ее современным и стильным дизайном, наличием готовых компонентов, хорошей документацией и поддержкой сообщества. Она позволит создать привлекательный и удобный пользовательский интерфейс для проекта.

1.6 Техническое задание

**Общие сведения**

Полное наименование работы: Разработка low-code платформы для создания ботов в мессенджере Telegram.

Исполнитель: Студент группы ИУК5-82Б Калужского филиала федерального государственного бюджетного образовательного учреждения высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» Фролов Кирилл Дмитриевич.

**Цели и назначение создания автоматизированной системы**

**Цель разработки**

Целью разработки является автоматизация создания ботов в мессенджере Telegram путем проектирования и реализации low-code платформы.

**Назначение системы**

Данная система предназначена для автоматизации создания ботов в мессенджере Telegram для различных сфер и услуг с помощью простого в использовании визуального конструктора. В настоящее время Telegram выступает не только в качестве мессенджера, но и в качестве платформы, где мелкие и средние бизнесы могут продавать свою продукцию или услуги через ботов или группы. Однако создание бота может занять много времени и требует опыта в программировании. Эта платформа решает эту проблему, позволяя любому человеку с базовыми знаниями в области компьютеров создавать своего собственного бота.

**Характеристика объектов автоматизации**

Объекты автоматизации для данной системы включают:

* Конструктор для создания ботов визуально понятным способом
* Модуль взаимодействия с Telegram API для отправки и получения сообщений и обработки команд
* Модуль отладки сформированных команд и создания файла с ботом
* Хранение данных о проектах и конфигурации в специальных файлах
* Механизмы валидации и обработки ошибок, позволяющие предотвратить некорректное поведение приложения
* Интерфейс для взаимодействия пользователя с системой, позволяющий создавать, редактировать и удалять проекты и команды ботов.
* Модуль интерпретации конструктора в код, который генерирует код на языке Python для создания ботов на основе пользовательских настроек и команд, заданных через конструктор.
* Возможность экспорта и импорта конфигурации и проектов для бэкапа или переноса между устройствами.

**Требования к автоматизированной системе**

**Требования к архитектуре**

Одним из требований к архитектуре проектируемой системы является использование MVVM (Model-View-ViewModel) паттерна проектирования [13]. Он обеспечивает разделение бизнес-логики и пользовательского интерфейса при разработке приложений с использованием технологии Windows Presentation Foundation (WPF).

Архитектура должна быть построена на основе принципа единственной ответственности (Single Responsibility Principle), что позволит сделать систему более расширяемой и удобной для поддержки.

В рамках данной архитектуры View не должен содержать никакой бизнес-логики и отвечает только за отображение данных, ViewModel выполняет задачи обработки данных и подготовки их для отображения в View, Model отвечает за доступ к данным и бизнес-логику.

Также необходимо обеспечить возможность использования DI-контейнера для внедрения зависимостей и обеспечения более гибкой конфигурации системы.

Приложение должно быть разбито на 3 слоя:

* Слой представления
* Слой бизнес-логики приложения
* Слой данных

**Функциональные требования**

Разработка графического интерфейса для создания чат-ботов

Возможность выбора из списка готовых блоков и элементов для создания бота

Возможность добавления новых блоков и элементов для создания бота

Возможность настройки параметров блоков и элементов через интерфейс

Возможность создания логики бота через блок-схему

Возможность запустить проект из системы

Возможность загрузки и хранения данных в файлах проекта

Возможность отправки сообщений от бота к пользователям

Возможность обработки сообщений от пользователей и выбор соответствующего ответа от бота

Возможность настройки параметров для интеграции с мессенджером Telegram

Возможность просматривать состояние бота при его работе

Возможность подключения и использования сторонних сервисов и API для расширения функционала

**Нефункциональные требования**

Система должна обеспечивать высокую производительность при работе с большим количеством пользователей

Система должна быть масштабируемой

Система должна быть безопасной и защищенной от несанкционированного доступа к данным

Система должна быть устойчивой к сбоям и восстанавливаться автоматически

Система должна быть удобной и интуитивно понятной для пользователей

**Состав и содержание работ по созданию автоматизированной системы**

|  |  |  |
| --- | --- | --- |
| Стадии | Этапы работ | Даты |
| 1. Формирование требований к АС 2. Разработка концепции АС 3. Техническое задание 4. Эскизный проект 5. Технический проект 6. Рабочая документация | * 1. Обследование объекта и обоснование необходимости создания АС   2. Формирование требований пользователя к АС   3. Оформление отчета о выполненной работе и заявки на разработку АС (тактико-технического задания)   4. Изучение объекта   5. Проведение необходимых научно-исследовательских работ   6. Разработка вариантов концепции АС и выбор варианта концепции АС, удовлетворяющего требованиям пользователя   7. Оформление отчета о выполненной работе   8. Разработка и утверждение технического задания на создание АС   9. Разработка предварительных проектных решений по системе и ее частям   10. Разработка документации на АС и ее части   5.1. Разработка проектных решений по системе и ее частям  5.2. Разработка документации на АС и ее части  5.3. Разработка и оформление документации на поставку изделий для комплектования АС и (или) технических требований (технических заданий) на их разработку  5.4. Разработка заданий на проектирование в смежных частях проекта объекта автоматизации  6.1. Разработка рабочей документации на систему и ее части  6.2. Разработка или адаптация программ | 23.07.2022 – 01.09.2022  01.09.2022 –  08.09.2022  08.09.2022 – 31.12.2022  01.01.2023 – 15.06.2023  15.06.2023-30.06.2023  01.07.2023 |

**Порядок разработки автоматизированной системы**

На этапе разработки концепций и изучения предметной области предъявляется научно-исследовательская часть расчетно-пояснительной записки.

На этапе технического задания предъявляется полноценное техническое задание, составленное в соответствии с необходимыми гостами.

На этапе эскизного проекта предъявляется эскизный проект, представляющий минимальную реализацию необходимого функционала и макеты пользовательских интерфейсов. Также предъявляется проектно-конструкторская часть расчетно-пояснительной записки.

На этапе технического проекта предъявляется готовый проект и проектно-технологическая часть расчетно-пояснительной записки.

На этапе рабочей документации предъявляются руководства пользователя, администратора и программиста.

**Порядок контроля и приемки системы**

По окончании работы предъявляется расчетно-пояснительная записка, в состав которой входят:

* Техническое задание;
* Научно-исследовательская часть;
* Проектно-конструкторская часть;
* Проектно-технологическая часть.

Так же должна быть предоставлена графическая часть работы, выполненная на листах в формате А1, в которую входят:

* Демонстрационные чертежи;
* Структурные схемы;
* Графики;
* Алгоритмические схемы.

**Требование к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие**

Подготовка аппаратной инфраструктуры, необходимой для работы системы,

Установка необходимого программного обеспечения, включая операционную систему, среду разработки и другие инструменты, используемые в процессе разработки и тестирования системы.

Разработка и тестирование приложений, использующих возможности системы low-code, в соответствии с требованиями.

Создание и документирование процессов, связанных с эксплуатацией и обслуживанием системы, включая процедуры резервного копирования, мониторинга и управления доступом.

Проведение тестирования и отладки системы, включая проверку работоспособности и надежности приложений, а также выполнение нагрузочных тестов.

Создание документации, содержащей описание архитектуры и функциональности системы, инструкции по ее эксплуатации и обслуживанию, а также справочные материалы для пользователей и разработчиков.

**Требования к документированию**

По окончанию работы предъявлена расчетно-пояснительная записка, в состав которой входят:

* техническое задание;
* научно-исследовательская часть;
* проектно-конструкторская часть;
* проектно-технологическая часть.

Также должна быть предоставлена графическая часть работы, выполненная на формате А1, в которую входят:

* демонстрационные чертежи;
* схема алгоритма
* структурная схема

**Источники разработки**

ГОСТ 34.601-90 «Информационная технология. Комплекс стандартов на автоматизированные системы. Стадии создания»;

ГОСТ 34.602-2020 «Информационная технология. Комплекс стандартов на автоматизированные системы».

2. ПРОЕКТНО – КОНСТРУКТОРСКАЯ ЧАСТЬ

2.1 Разработка структуры приложения

Система имеет 3 уровневую архитектуру с применением шаблона MVVM.

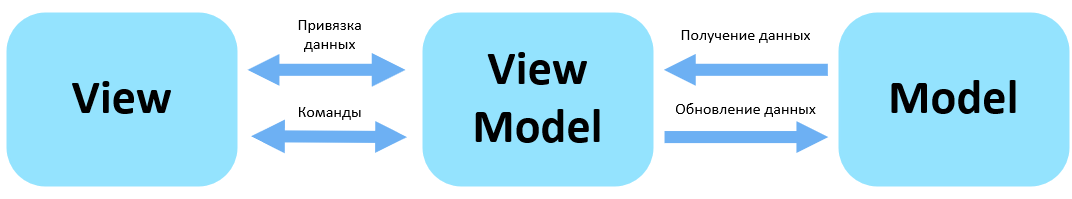


Рисунок 2.1.1 – Шаблон MVVM

Архитектура приложения строго разделена на три уровня, каждый из которых выполняет свою роль. На первом уровне расположен слой представления, который представляет данные в виде конфигурационных файлов проекта, моделей и объектов. При любом изменении этих данных, представление автоматически обновляется благодаря модели событий. Такой подход обеспечивает высокую согласованность данных.

Важно отметить, что изменение интерфейса не повлечет за собой необходимость переделывания всего проекта. Это связано с тем, что представление не может изменять данные или выполнять с ними какие-либо действия. Таким образом, изменения, связанные с представлением, могут быть реализованы независимо от изменений на более низких уровнях. Это обеспечивает более гибкое и эффективное управление проектом.

Слой модели представления является связующим звеном между данными и интерфейсом приложения. В этом слое реализован мост между данными и интерфейсом, где привязки данных и поведение при различных действиях обеспечивают целостность полученных данных. В приложении, слой модели представления управляет поведением главного окна, созданием проекта и конструктором бота. Модель бота, его поведение, генерирование функций и информация о проекте также реализованы в этом слое. Взаимодействие с различными модулями и командами, а также выгрузка и загрузка данных осуществляются в этом слое. Такой подход позволяет обеспечить целостность данных и динамическое обновление их состояния, а также избежать необходимости изменения всего проекта при изменении интерфейса.

Слой модели представляет собой важный компонент приложения, который содержит не только модель бота и его поведение, но также включает в себя хранение и обработку данных о пользователях, статистику использования бота, настройки для работы с API мессенджера и другие вспомогательные модули. Кроме того, в слое модели реализована логика взаимодействия с внешним модулем целостности системы, что позволяет расширять функциональность бота и повышать его эффективность в работе.

Пользователи могут создавать неограниченное количество команд для своего чат-бота в рамках допустимого для API Telegram. Команды отображаются в виде списка, и при выборе одной из команд, становятся доступными настройки для управления поведением бота. Настройки команд являются основным инструментом для создания логики бота, пользователи могут добавлять файлы, изображения и привязывать к ним кнопки. Для проверки работы бота предусмотрена функция отладки, которая позволяет пользователям запустить бота и проверить его функциональность. После того, как проект будет завершен, пользователи могут сформировать исполняемый файл для удобного запуска и остановки бота.

Важным аспектом архитектуры данного проекта является ее гибкость и модульность. Разделение на три слоя позволяет легко менять и расширять функциональность бота, не затрагивая остальные компоненты. Кроме того, наличие древовидной структуры команд и кнопок позволяет удобно организовывать логику работы бота и создавать сложные сценарии взаимодействия с пользователем [8]. В целом, архитектура проекта обеспечивает высокую гибкость, расширяемость и удобство в использовании для разработчиков и конечных пользователей.

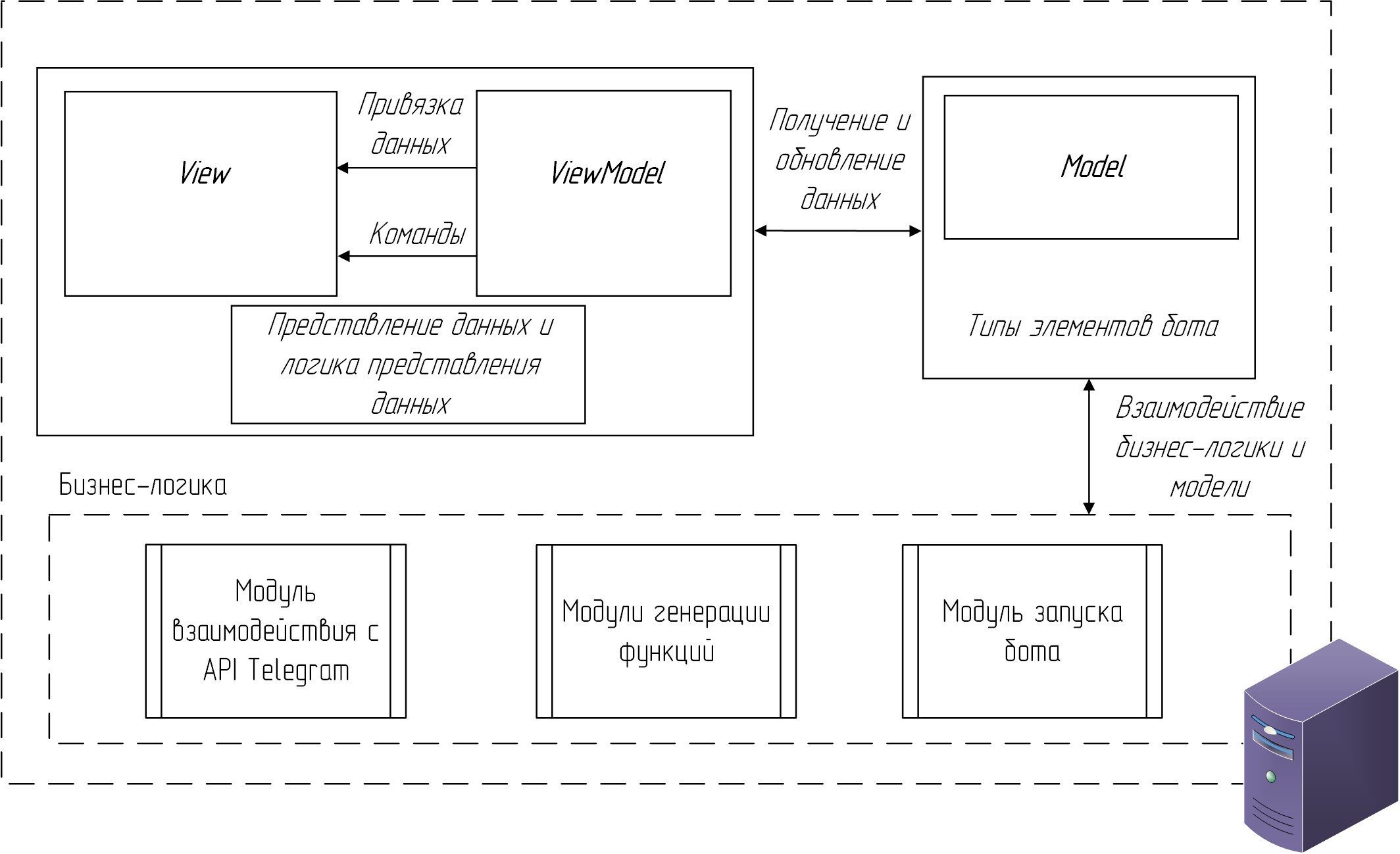


Рисунок 2.1.2 – Схема архитектуры приложения

Для описания взаимодействия пользователей с системой была составлена UML-диаграмма прецедентов (см. Приложение А)

Команды и кнопки представляют собой основные элементы управления чат-ботом. Они могут использоваться для определения логики работы бота и отправки различных мультимедийных данных в ответ на действия пользователя.

Команды представляют из себя корневой элемент структуры, и могут быть выполнены пользователем в любой момент. Каждая команда содержит в себе набор параметров, которые могут быть переданы боту при ее выполнении.

Кнопки в свою очередь делятся на Inline-кнопки и Markup-кнопки, они могут быть добавлены в ответ на команду, либо как реакция на ввод пользователя. Обе кнопки имеют древовидную структуру, где каждая кнопка может иметь несколько дочерних элементов. Однако, в отличие от команд, кнопки не могут быть выполнены пользователем напрямую.

Inline-кнопки представляют собой кнопки, которые находятся внутри сообщения бота и могут быть нажаты пользователем. Каждая Inline-кнопка содержит в себе уникальный идентификатор, который может быть использован для определения реакции на нажатие этой кнопки.

Markup-кнопки представляют собой кнопки, которые расположены под строкой ввода сообщения, при нажатии на кнопку ее название отправляется как сообщение чат боту. Чтобы пользователь смог использовать их ему необходимо отправить команду или воспользоваться кнопкой, это будет зависеть от логики самого чат бота. Каждая Markup-кнопка также содержит в себе уникальный идентификатор, который может быть использован для определения реакции на выбор этой кнопки.

Для обеспечения взаимодействия с Telegram в проекте была принята решение использовать библиотеку aiogram. Эта библиотека является одной из наиболее популярных и мощных библиотек на языке Python для работы с API Telegram. Установка библиотеки aiogram осуществляется с помощью инструмента управления пакетами Python - pip. Также было принято решение включить саму библиотеку aiogram в состав программы в виде архива, который поставляется вместе с программой. Это позволяет обеспечить удобство развертывания и использования библиотеки без необходимости отдельной установки ее через pip на каждой системе, где будет запускаться программа.

2.2 Диаграмма классов

Диаграмма классов (Рисунок 2.2.1) отображает бизнес-логику проекта и представляет структуру классов, их взаимосвязи и основные функциональные элементы системы.

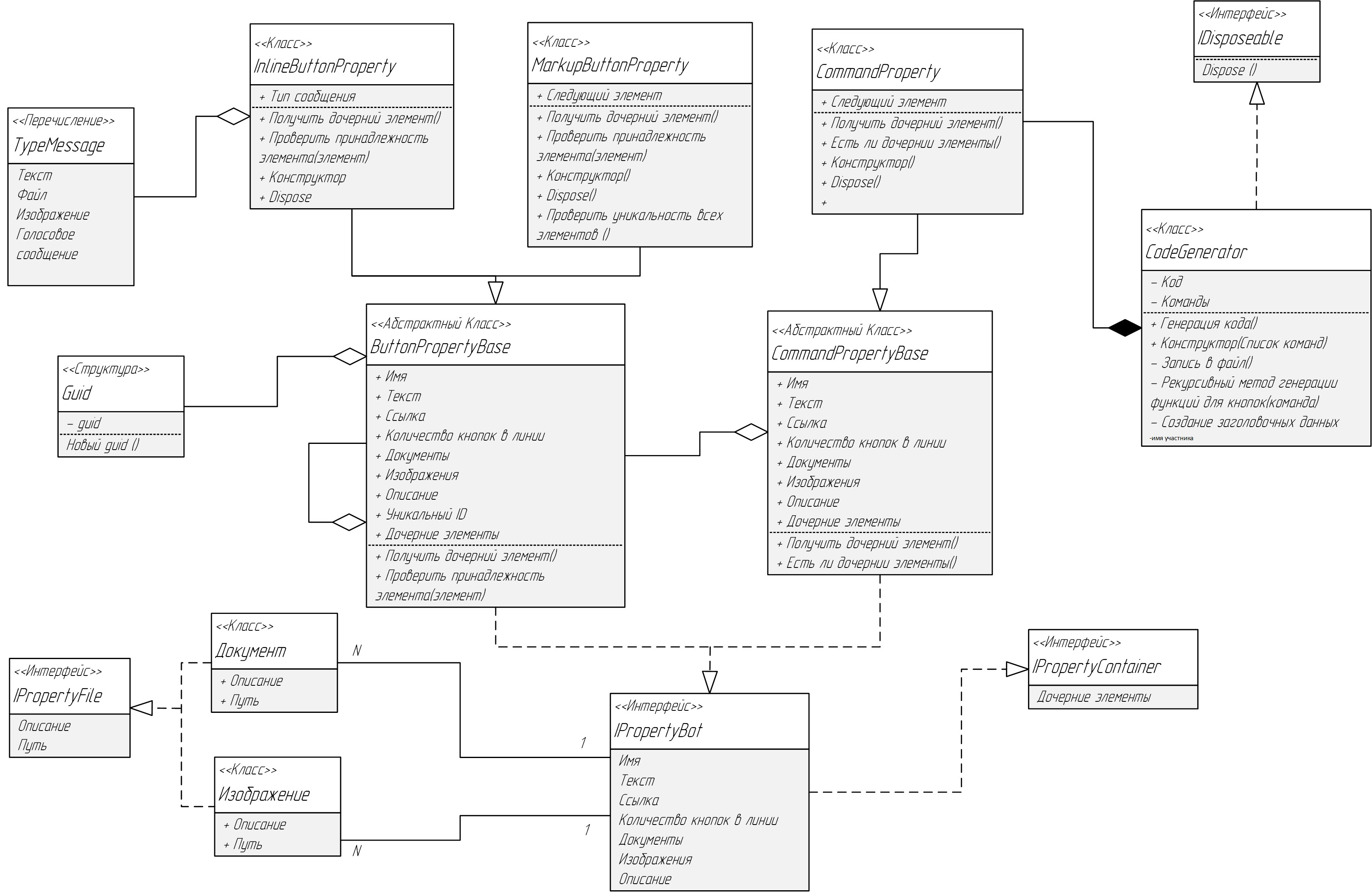


Рисунок 2.2.1 – Диаграмма классов

2.3 Разработка алгоритмов обработки информации

Для реализации функции генерации кода было принято решение использовать файл ресурсов, содержащий заранее подготовленные шаблоны кода. Затем алгоритм формирует код, основываясь на этих шаблонах и введенных пользователем данных. В процессе алгоритм перебирает все непустые команды и их параметры, заменяя ключевые слова в шаблоне соответствующими данными.

Так как этот процесс может занимать значительное время, особенно при обработке большого количества команд и взаимодействии с файлом (чтение и запись), было принято решение выполнить его в отдельном потоке[7]. Это предотвращает блокировку пользовательского интерфейса и некорректное завершение программы.

На основе указанных условий была разработана блок-схема алгоритма, а затем был реализован алгоритм. Он позволяет генерировать код эффективно и без проблем с интерфейсом, обеспечивая корректное выполнение программы.

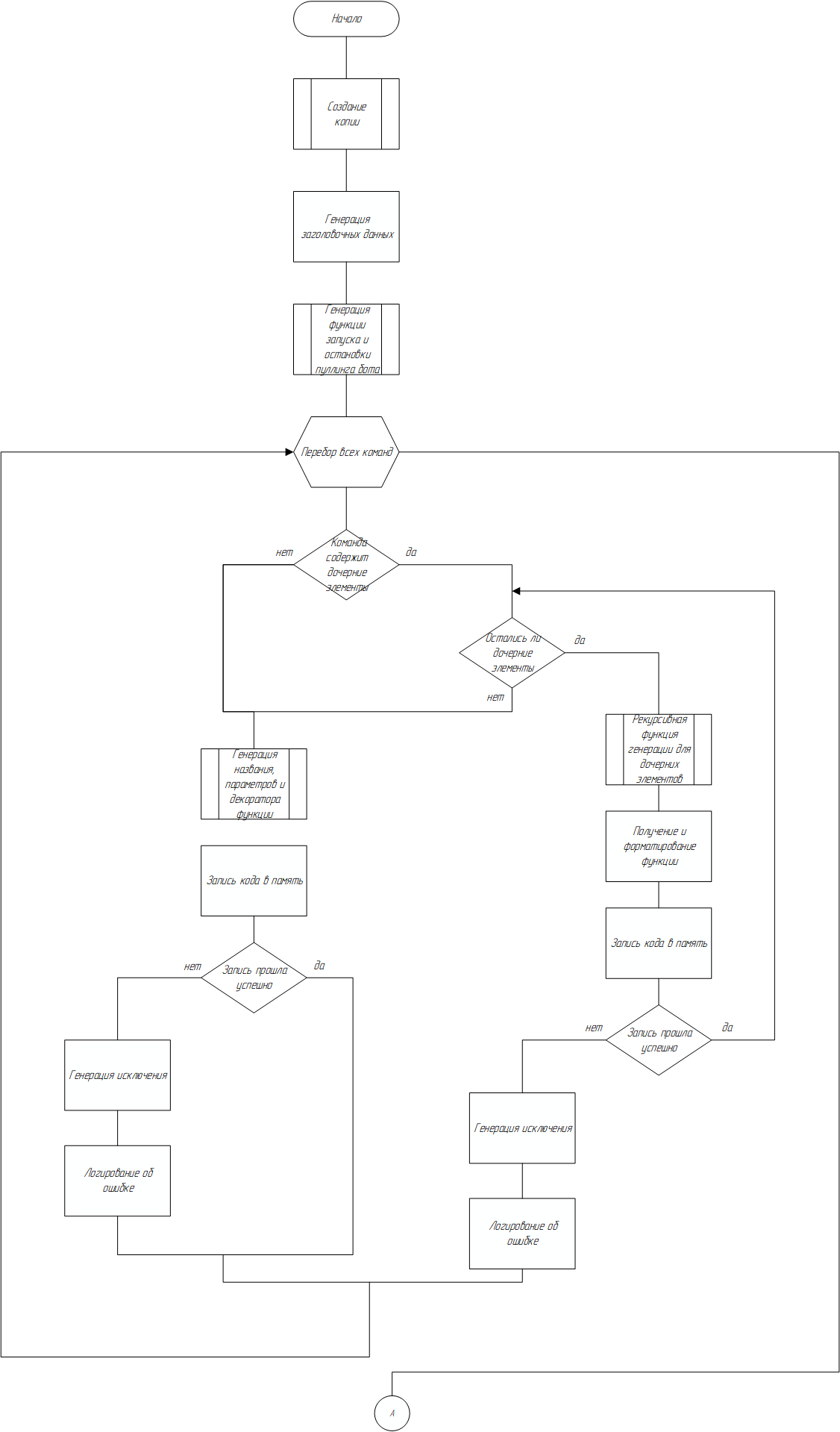


Рисунок 2.3.1 – Алгоритм генерации функций 1 часть

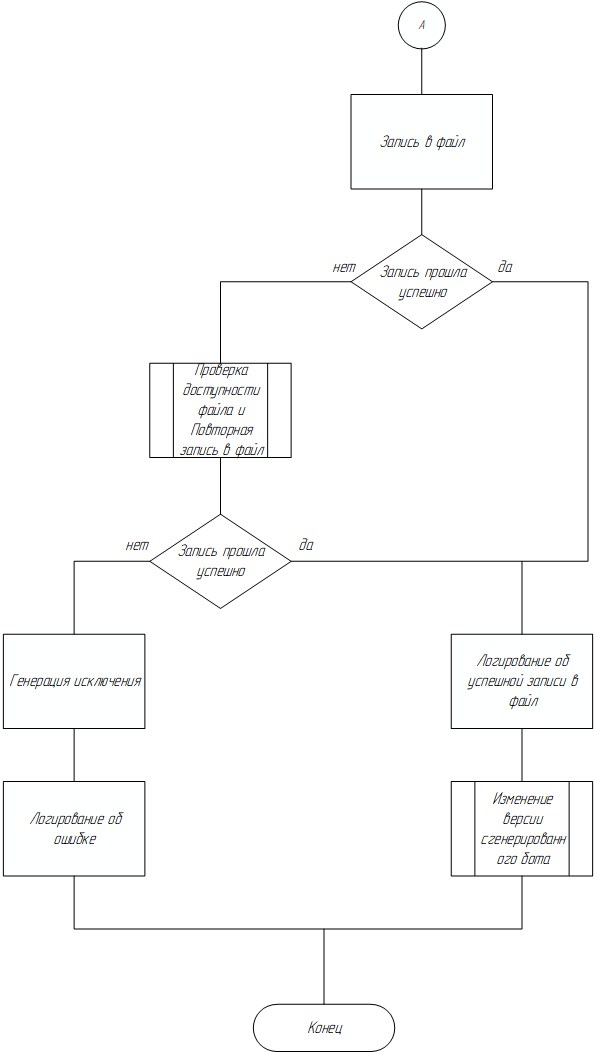


Рисунок 2.3.2 – Алгоритм генерации функций 2 часть

Алгоритм создания проекта является одним из ключевых компонентов приложения, поскольку он отвечает за формирование основных данных, необходимых для создания функций и самого чат-бота.

В этом алгоритме осуществляется создание конфигурационного файла, а также выполняется логирование практически на каждом этапе. Логирование играет важную роль, так как в случае некорректного создания проекта оно помогает определить точку и причину возникновения ошибки. Это особенно важно в данном алгоритме.

Формирование конфигурационного файла происходит только при наличии всех необходимых данных, что помогает избежать лишней нагрузки на систему [3]. В случае неудачной записи конфигурационного файла предусмотрена повторная попытка записи с повторным формированием файла. Если формирование конфигурационного файла не удалось, проект не будет инициализирован, чтобы избежать неправильной работы системы.

Таким образом, алгоритм создания проекта играет важную роль в обеспечении корректного и надежного функционирования приложения.

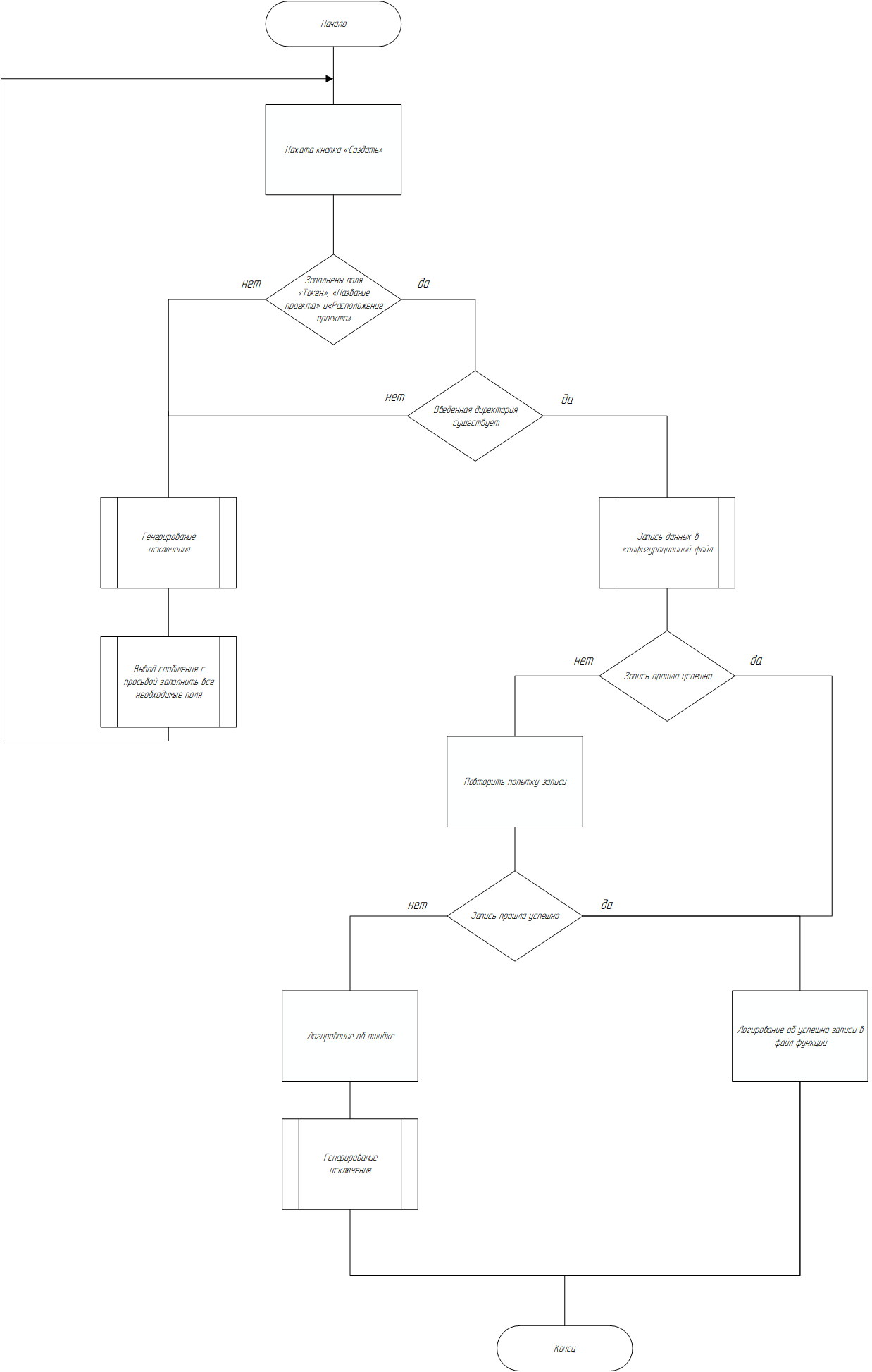


Рисунок 2.3.3 – Алгоритм создания проекта

Алгоритм проверки целостности системы выполняет проверку необходимых зависимостей, таких как наличие установленного Python, наличие библиотеки aiogram через pip и настройку рабочей среды. В случае отсутствия определенных зависимостей, алгоритм автоматически устанавливает их и создает необходимую среду для работы системы.

Кроме того, текущее состояние системы хранится в конфигурационном файле, что позволяет более эффективно мониторить и контролировать работу системы. Конфигурационный файл содержит информацию о настройках, параметрах и других важных данных, которые используются в процессе работы программы.

Этот алгоритм проверки целостности системы гарантирует, что все необходимые зависимости установлены и настроены корректно, что является важным шагом перед запуском программы. Он обеспечивает надежность и стабильность работы системы, а также упрощает процесс установки и настройки для пользователей.

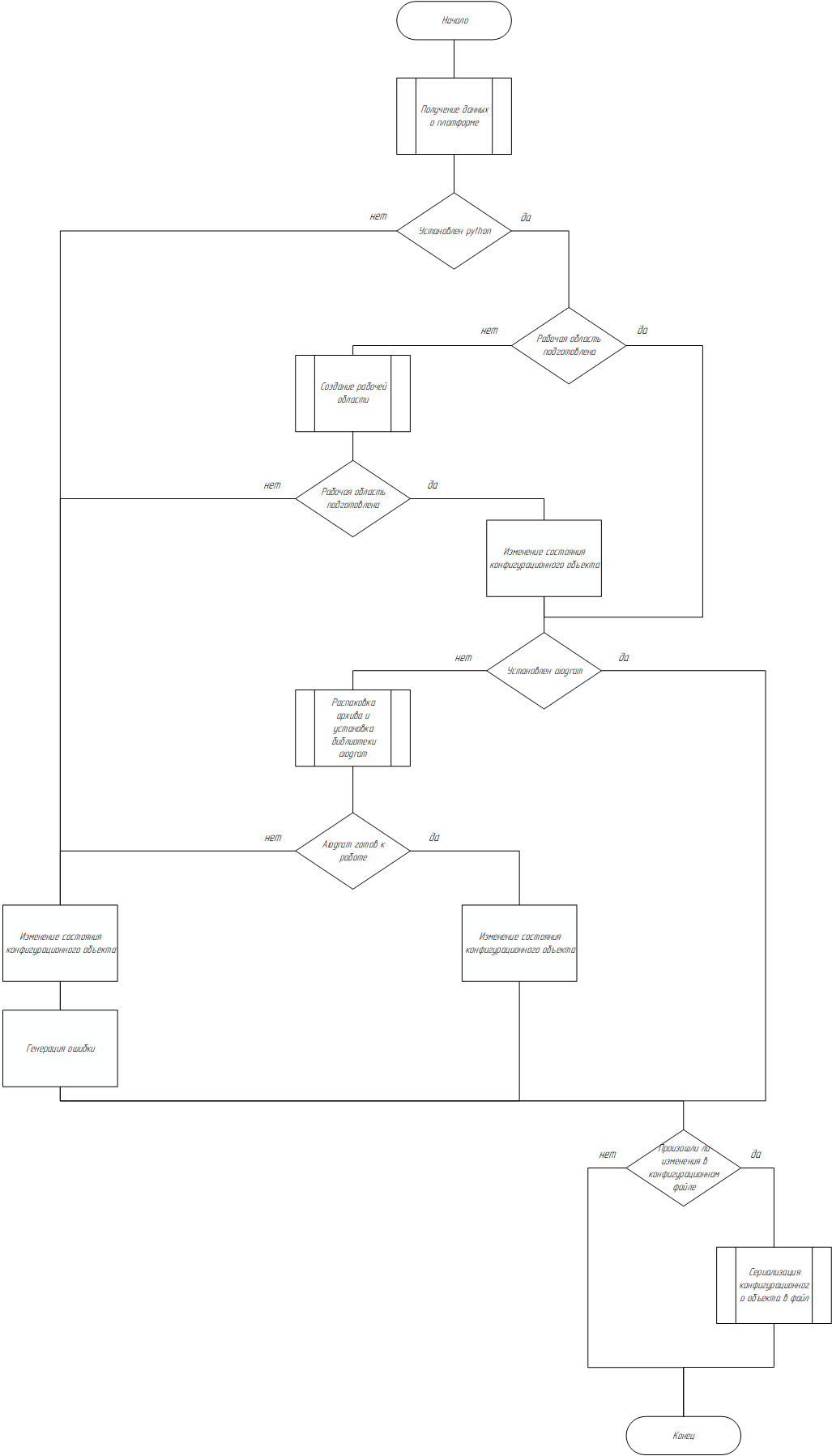


Рисунок 2.3.4 – Алгоритм проверки целостности системы

На рисунке 2.3.5 представлена диаграмма классов, которая отображает структуру классов, отвечающих за сериализацию объектов телеграм-бота. Данная диаграмма представляет набор классов, которые выполняют задачи по преобразованию объектов телеграм-бота в формат, пригодный для сохранения или передачи, а также обратное преобразование этих данных[12].

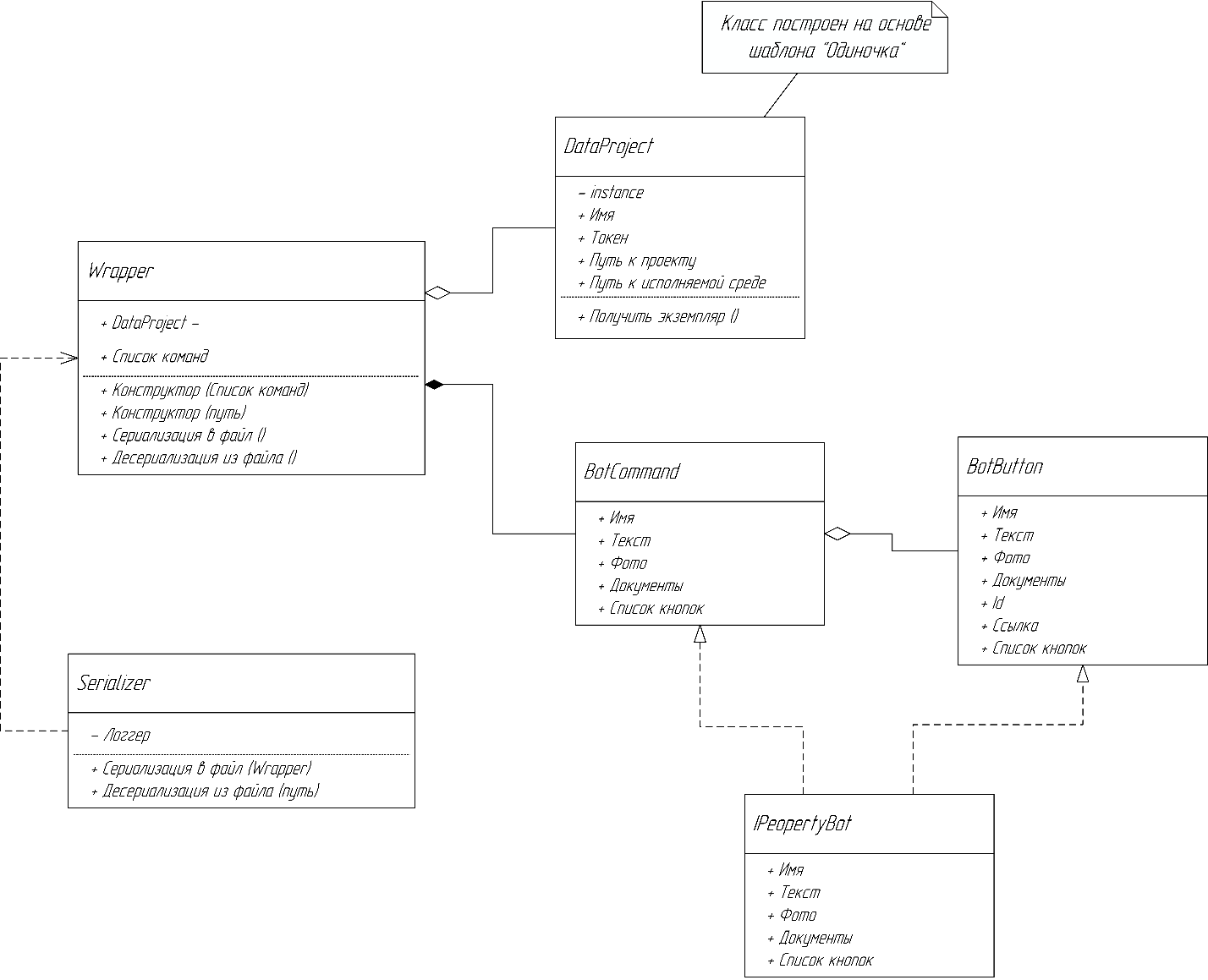


Рисунок 2.3.5 – Диаграмма классов отвечающих за сериализацию

На рисунке 2.3.6 представлен алгоритм сериализации кнопок в виде рекурсивной функции. Данный алгоритм позволяет обработать все элементы дерева, сериализуя каждый созданный объект [14].

Алгоритм начинает с корневого элемента дерева, который представляет кнопку, и рекурсивно проходится по всем его дочерним элементам. Для каждого элемента, алгоритм выполняет сериализацию, преобразуя объект в формат, пригодный для сохранения или передачи данных.

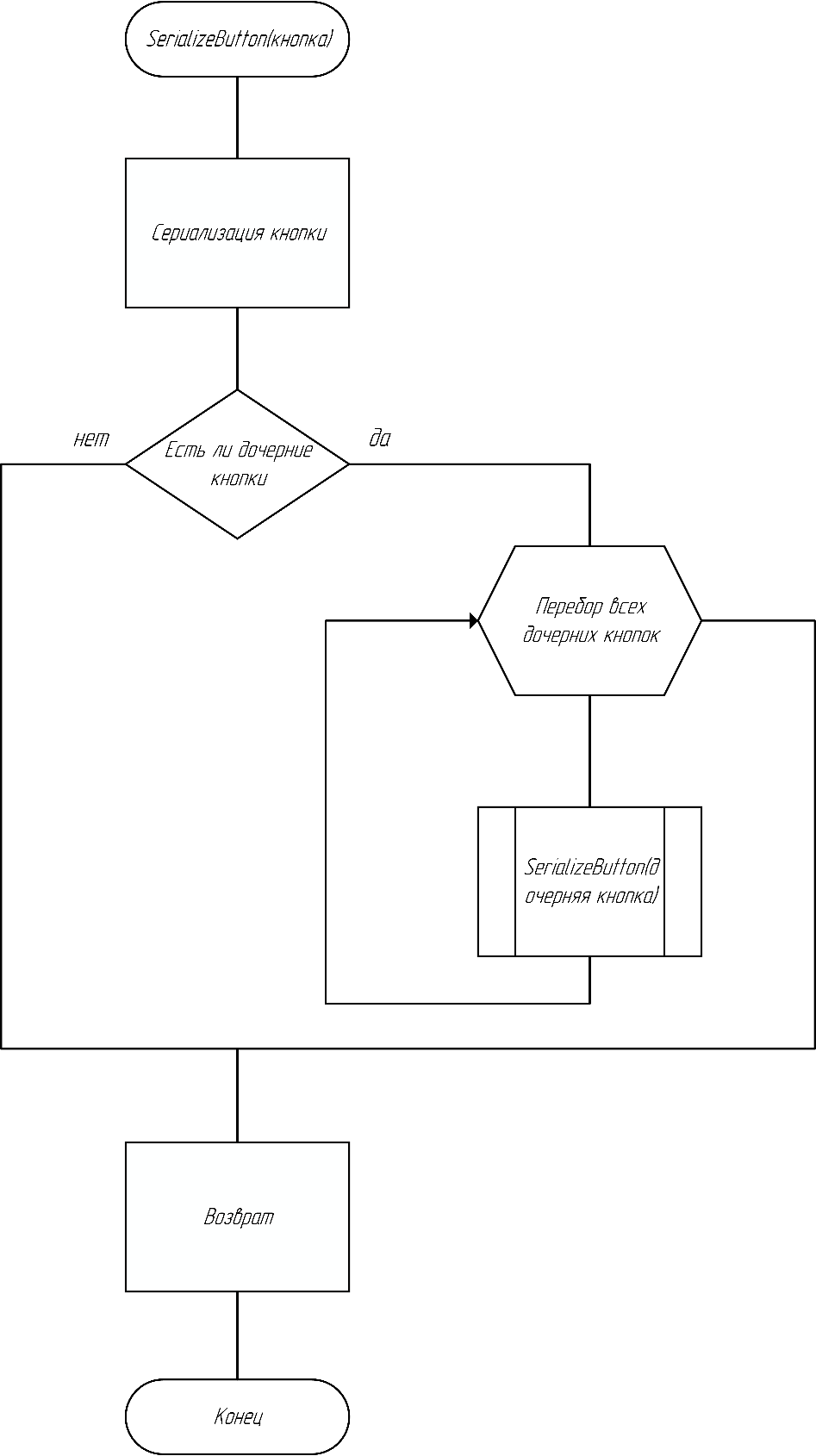


Рисунок 2.3.6 – Рекурсивная функция сериализации объектов

Пример XML-файла с сериализованными данными Листинг 2.3.1 –

XML-файл.

<?xml version="1.0" encoding="utf-8"?>

<WrapperDataBot xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<DataProject>

<Name>Test</Name>

<PathDirectory></PathDirectory>

<Path></Path>

<PathEnvironment></PathEnvironment>

<DateProject></DateProject>

</DataProject>

<CommandProperties>

<BotCommandProperty>

<Id>0</Id>

<Name>start</Name>

<CountButtonInLine>0</CountButtonInLine>

<Description>Начальная команда</Description>

<Photos />

<Documents />

<Children>

<ButtonBotBase xsi:type="InlineButtonProperty">

<Name>Inline Button</Name>

<Photos />

<Documents />

<UniqueId>0f52833a-6e43-46a0-bf16-4e507e372c91</UniqueId>

<CountButtonInLine>0</CountButtonInLine>

<Children />

</ButtonBotBase>

<ButtonBotBase xsi:type="MarkupButtonProperty">

<Name>Markup Button</Name>

<Photos />

<Documents />

<UniqueId>d16ae2c2-8643-436f-a044-cc8dd4b8a21c</UniqueId>

<CountButtonInLine>0</CountButtonInLine>

<Children />

</ButtonBotBase>

</Children>

</BotCommandProperty>

</CommandProperties>

</WrapperDataBot>

В данном XML представлении содержатся сериализованные данные о проекте. Он состоит из двух основных блоков: <DataProject>, в котором содержатся общие данные о проекте, и <CommandProperties>, где расположены элементы управления чат-ботом.

В блоке <DataProject> хранятся информация, относящаяся к проекту в целом, такие как его название, путь к проекту, путь к исполняемой среде и дата создания. Этот блок предоставляет общую контекстную информацию о проекте.

В блоке <CommandProperties> находятся все элементы управления чат-ботом. Команды представлены тегом <BotCommandProperty> и содержат информацию о команде, такую как ее имя и описание. Кнопки представлены тегами <ButtonBotBase xsi:type="InlineButtonProperty"> и <ButtonBotBase xsi:type="MarkupButtonProperty">, и у каждой кнопки есть свои уникальные свойства и параметры, которые определяют ее поведение.

Такой формат XML-файла обеспечивает структурированное хранение сериализованных данных о проекте и его элементах управления. Он позволяет легко считывать и обрабатывать эти данные в приложении для реализации необходимой логики и взаимодействия с чат-ботом.

2.4 Разработка интерфейса взаимодействия пользователя с системой

В ходе проектирования программного интерфейса, было решено, что он будет состоять из 2 главных окон. Первое окно должно содержать список всех созданных проектов, и краткую информацию о них [9]. Это позволит быстро запускать нужные проекты. Также в этом окне должно быть реализовано создание и открытие проекта. Исходя из вышеперечисленных условий был разработан макет интерфейса, он демонстрируется на рисунке 2.4.1

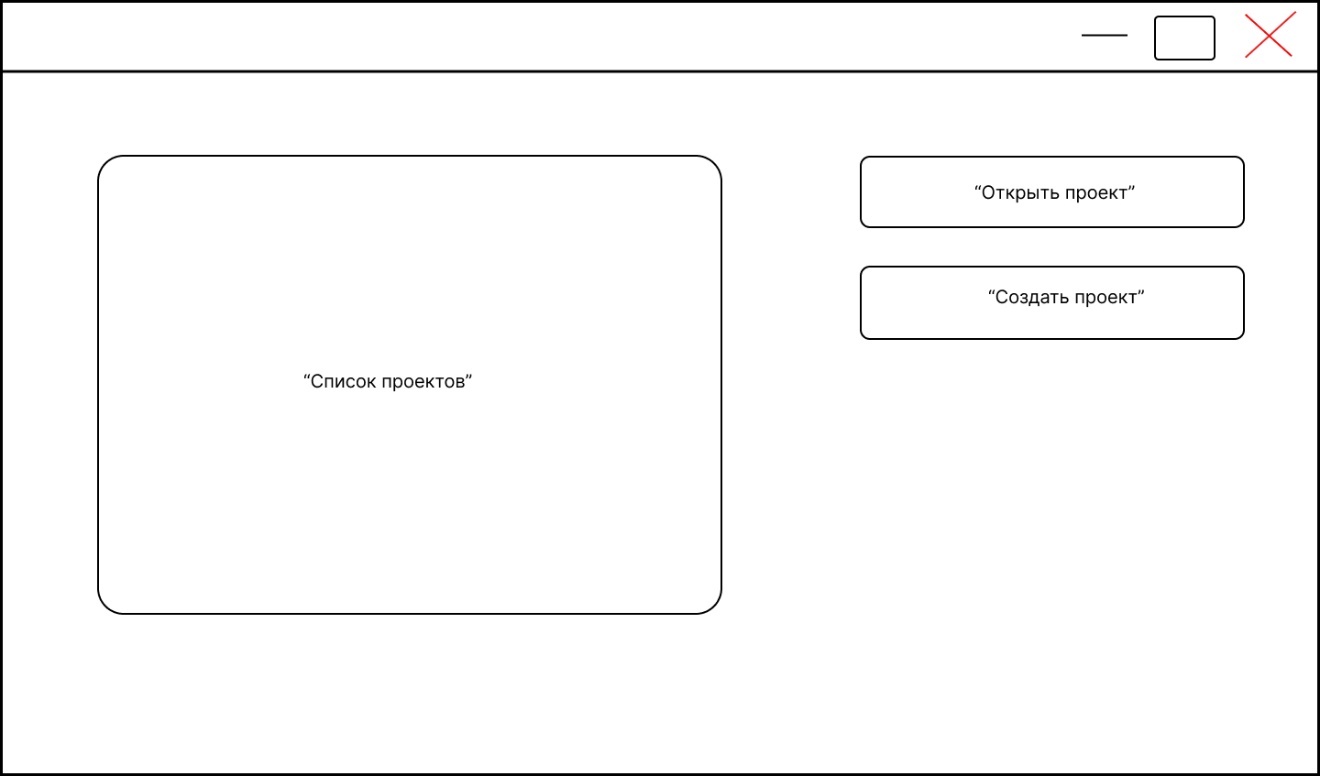


Рисунок 2.4.1 – Макет окна «Меню проектов»

Для создания проекта был разработан макет окна позволяющий легко создавать новые проекты в системе. Для создания проекта необходимо заполнить необходимые параметры, которые будут использоваться системой при создании бота. Интерфейс создания проекта был разработан с учетом простоты использования и интуитивно понятного интерфейса, что позволяет даже новичкам легко создавать новые проекты в системе. Рисунок 2.4.2 демонстрирует, как пользователь может заполнить необходимые параметры перед созданием проекта.

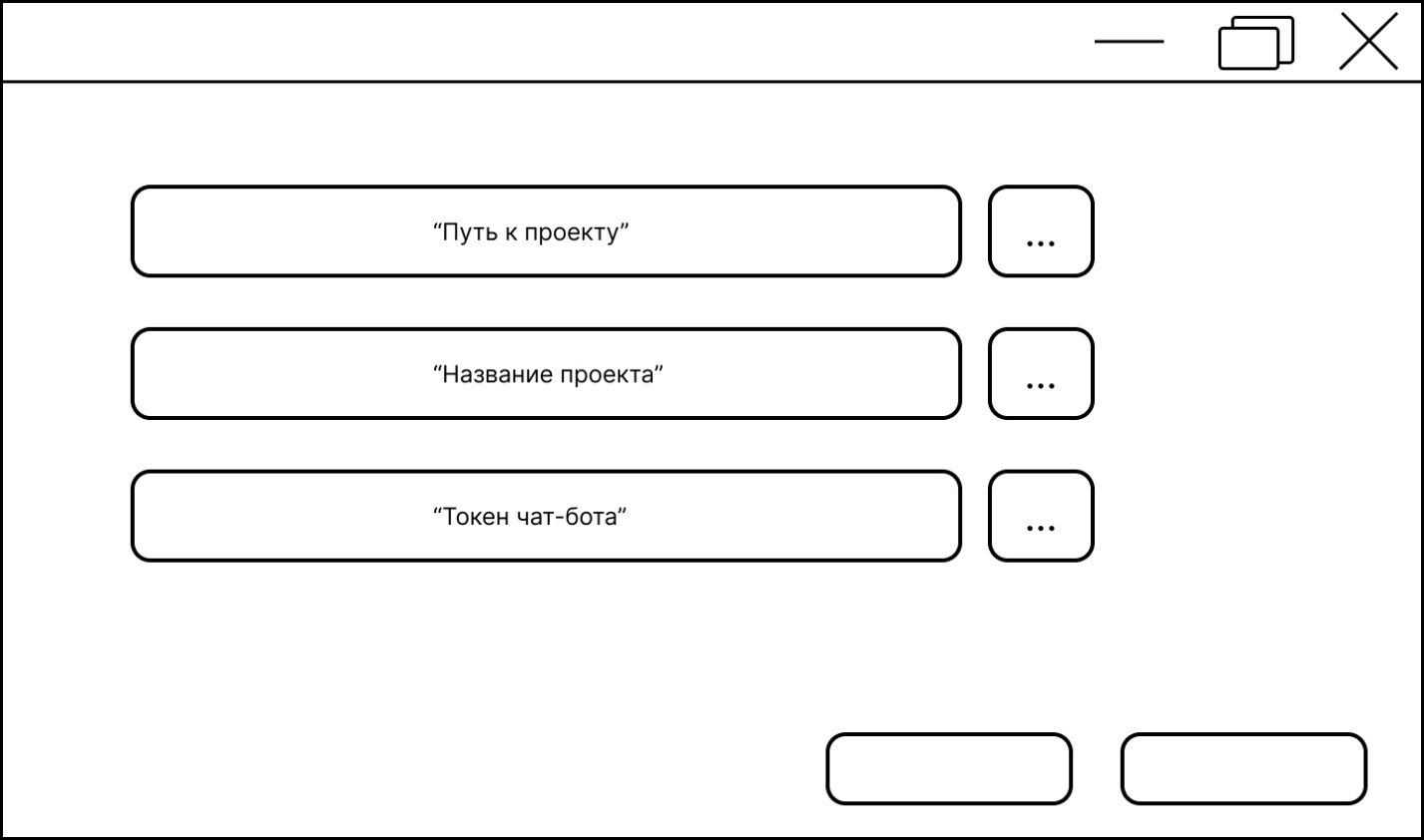


Рисунок 2.4.2 – Макет окна создания проекта

После выбора или создания проекта, открывается окно с конструированием бота. В этом окне должна происходить разработка команд, кнопок и другого дополнительного контента для бота. Интерфейс данного окна должен содержать быстрый доступ к элементам бота, рабочую область, где будут представлены команды и кнопки, визуализация дерева объектов, и настройка этих элементов. Также для удобной работы с проектом, должно быть реализовано меню, в котором будут располагаться различные функции проекта, такие как сохранить проект, создать проект и похожие функции.



Рисунок 2.4.3 –Окно конструирования чат-бота.

Окно управления чат-ботом - это необходимый компонент платформы (Рисунок 2.4.4), представляющий собой удобный интерфейс для управления работой бота. Оно должно быть оберткой над консолью, что позволяет легко управлять ботом и просматривать основные параметры, включая текущее состояние, статус подключения и другие важные данные, отображаемые на главном экране.

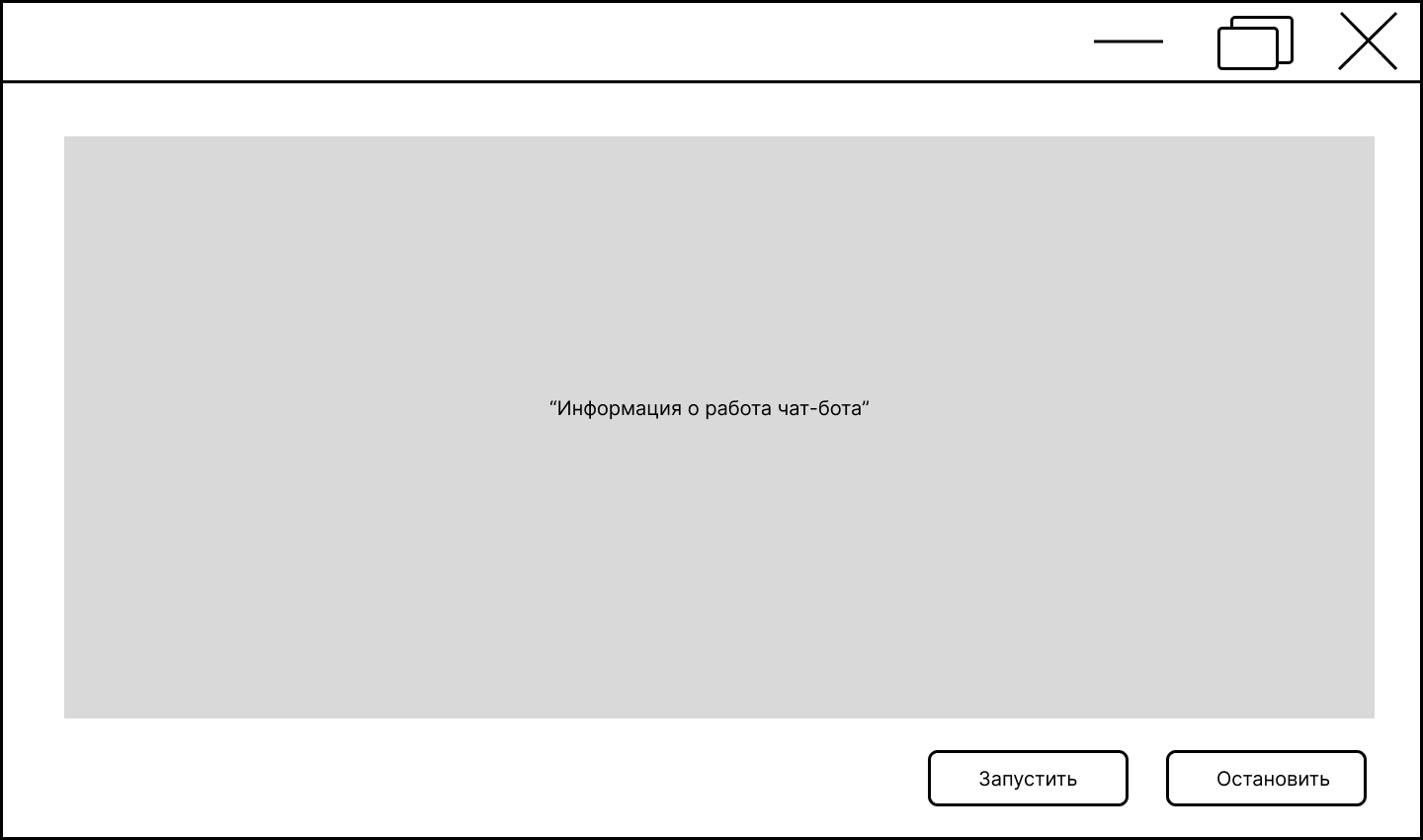


Рисунок 2.4.4 – Макет окна управления чат-ботом.

Все представленные макеты интерфейса должны быть адаптированы под разные дисплеи. На рисунке 2.4.5 представлен макет примера адаптации окна для конструирования бота.

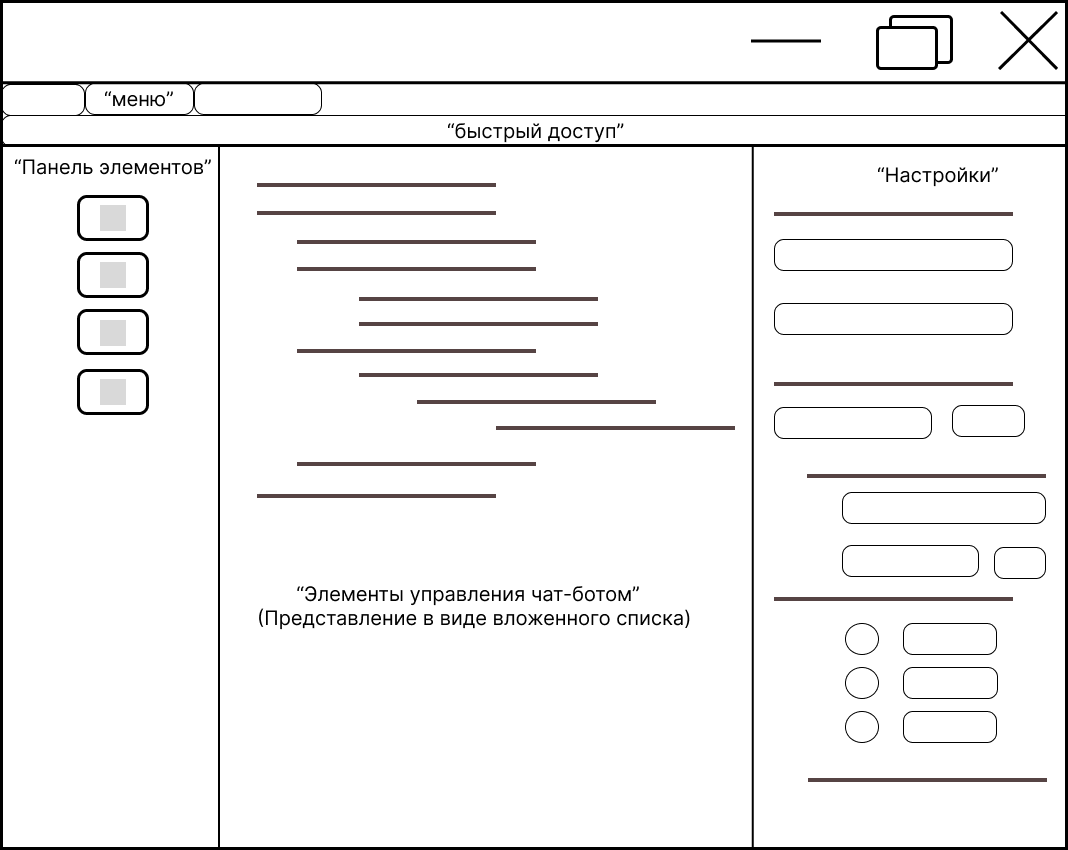


Рисунок 2.4.5 – Макет адаптации окна для разных дисплеев.

3. ПРОЕКТНО – ТЕХНОЛОГИЧЕСКАЯ ЧАСТЬ

3.1 Тестирование и отладка рабочей программы

Для тестирования приложения были созданы тест-кейсы основных окон и страниц приложения.

Таблица 3.1.1 – Тест-кейс главного окна

|  |  |
| --- | --- |
| Номер | 1 |
| Заголовок | Меню проектов |
| Предусловие | Запустить приложение |
| Шаг | Ожидаемый результат |
| Выбрать в представленном списке проект | Открытие нового окна с проектом |
| Нажать на кнопку «Открыть проект» | Запуск модального окна с выбором файла проекта |
| Нажать на кнопку «Создание проекта» | Запуск окна «Создание проекта» |

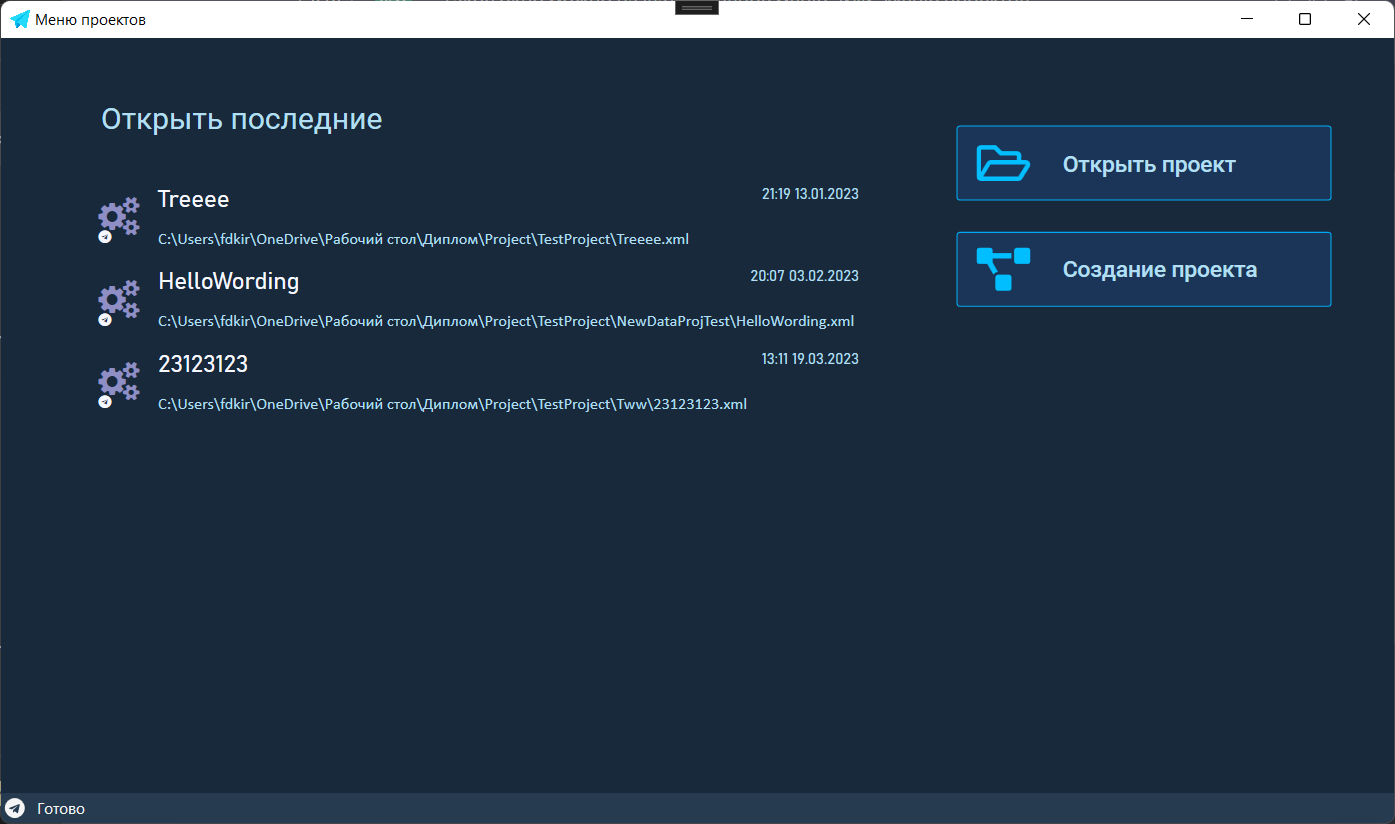


Рисунок 3.1.1 – Окно меню проектов

Таблица 3.1.2 – Тест-кейс окна регистрации

|  |  |
| --- | --- |
| Номер | 2 |
| Заголовок | Создание проекта |
| Предусловие | Нажать на кнопку «Создание проекта» в главном окне |
| Шаг | Ожидаемый результат |
| Нажать на кнопку «Создать» без введенных нужных данных | Сообщение с просьбой заполнить нужные поля |
| Заполнить все поля и нажать на кнопку «Создать» | Запуск окна «Конструктор бота» |
| Нажать на кнопку «…» | Запуск модального окна с выбором директории |
| Нажать на кнопку «Отмена» | Возврат к окну «Главное окно» |

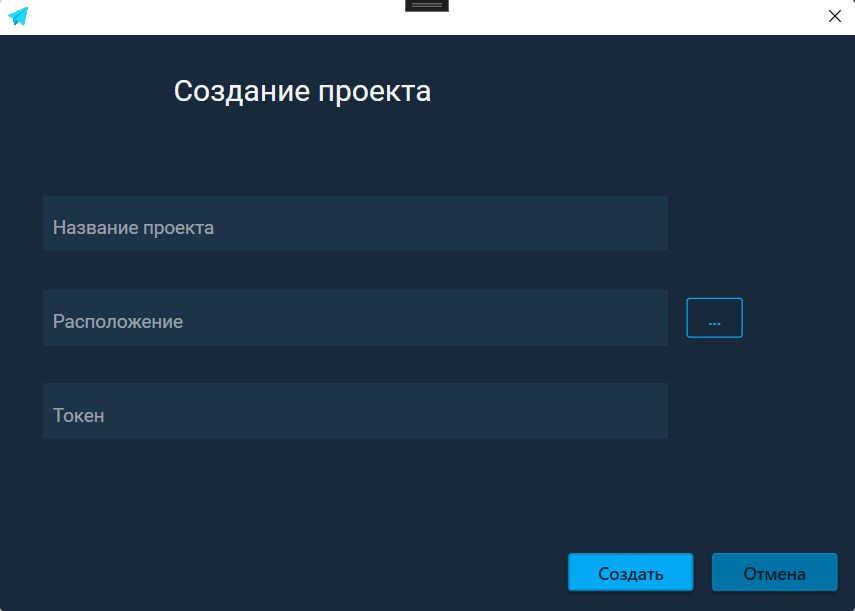


Рисунок 3.1.2 – Окно создания проекта

Таблица 3.1.3 Тест-кейс окна конструирование бота

|  |  |
| --- | --- |
| Номер | 3 |
| Заголовок | Конструктор бота |
| Предусловие | Введены корректные данные в окне «Создание проекта» и нажата кнопка «Создать» |
| **Шаг** | **Ожидаемый результат** |
| Нажать на кнопку «Команда» или «Текст» | Добавление в список по центру новой команды или текста |
| Нажать на кнопку «Изменить токен» | Запуск модального окна «Изменение токена» |
| Нажать на команду в списке | Представление в блоке «Настройки» основного функционала команды |
| Нажать на кнопку «Создать» | Запуск окна с работой сконструированного бота |
| Нажать на кнопку «…» | Запуск модального окна с выбором файла |
| Изменение настроек | Отображение в нижнем левом углу статуса изменения |

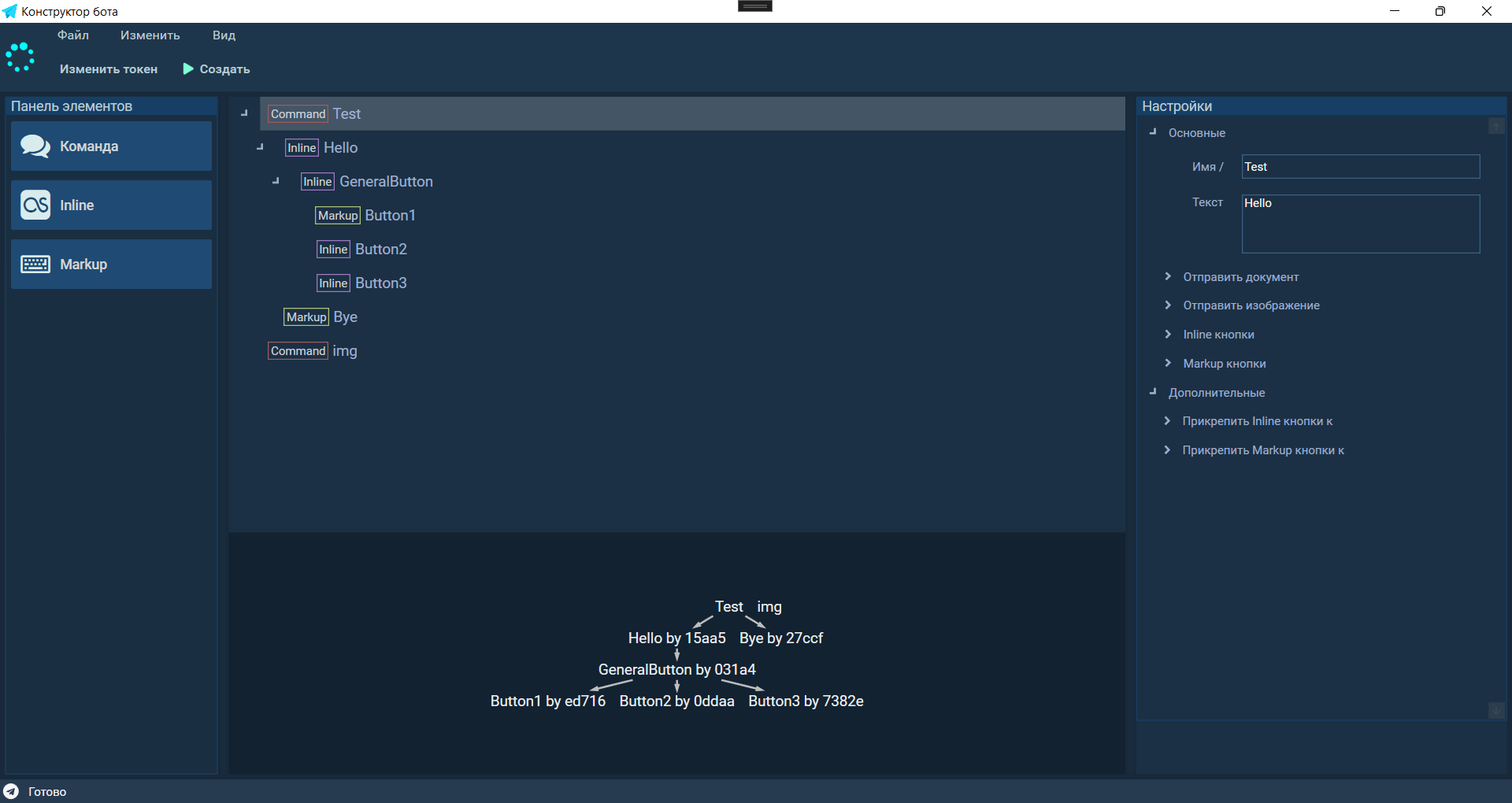


Рисунок 3.1.3 – Окно конструирование бота

Таблица 3.1.4 Тест-кейс результата конструктора

|  |  |
| --- | --- |
| Номер | 4 |
| Заголовок | Окно состояния бота |
| Предусловие | Созданы команды и нажата кнопка «запустить» |
| **Шаг** | **Ожидаемый результат** |
| Нажать на кнопку «Запустить» | Кнопка должна быть недоступна |
| Нажать на кнопку «Остановить» | Появляется шестеренка, означающая выключение бота, следом должно вывестись сообщение «Бот остановлен» |
| Нажата кнопка закрытия окна | Появляется шестеренка означающая выключение бота, следом закрытие окна |
| Открытие окна | Вывелось сообщение «Запуска бота» и «Бот запущен» |

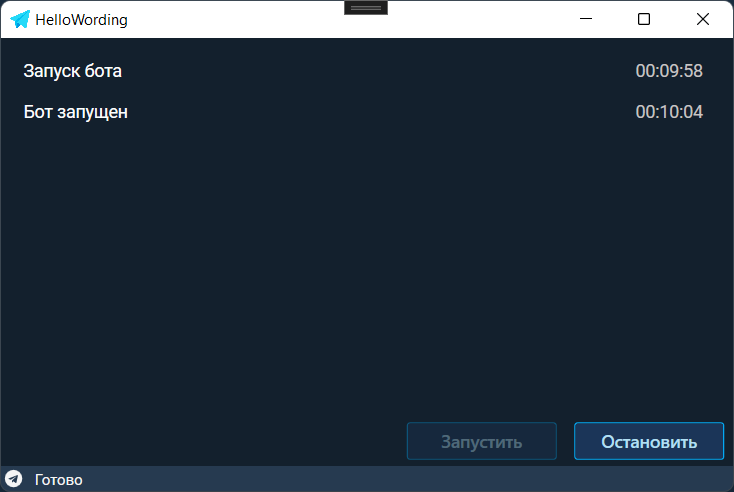


Рисунок 3.1.4 – Окно конструирование бота

Таблица 3.1.5 Тест-кейс результата конструктора

|  |  |
| --- | --- |
| Номер | 5 |
| Заголовок | Работа чат-бота |
| Предусловие | Статус «бот запущен» |
| **Шаг** | **Ожидаемый результат** |
| Отправлена созданная команда боту | Ответ должен представлять введенные данные в конструктора в виде текста или файлов |
| Отправлена команда, не предусмотренная в конструкторе | Отсутствие сообщений |
| Выбор документа | Ответ должен состоять из документа и подписи к нему, если подпись не указана приходит только документ |
| Выбор изображения | Ответ должен состоять из изображения и подписи к нему, если подпись не указана приходит только изображение |

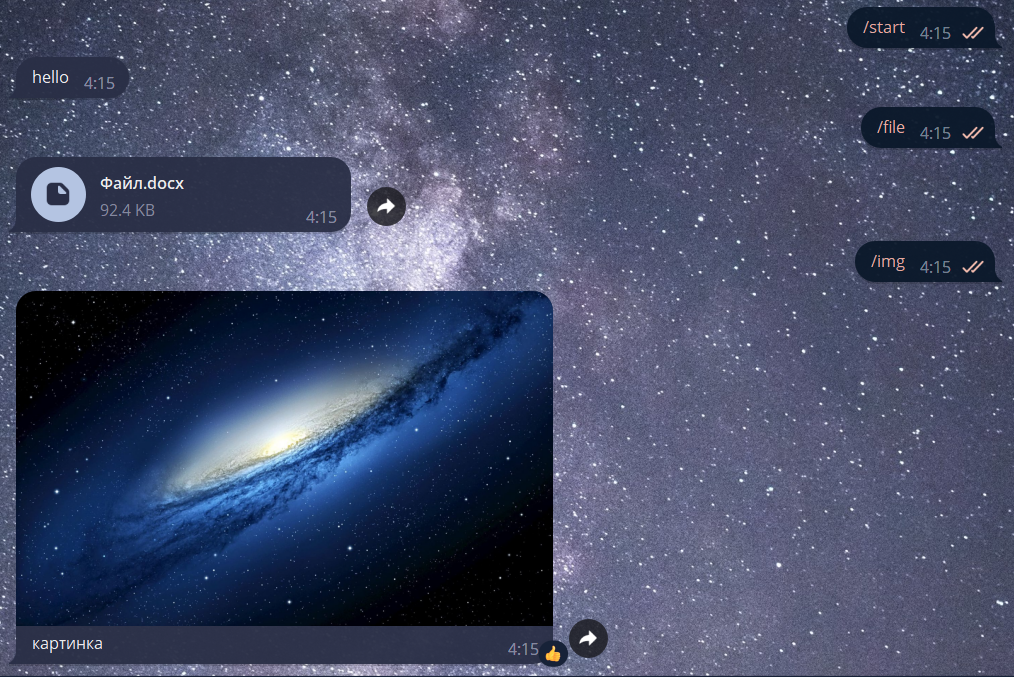


Рисунок 3.1.5 – Ожидаемый результат сконструированного бота

3.2 Порядок развертывания системы

**Подготовка окружения**

Для работы системы необходимо заранее установить следующие компоненты: .Net 5 и Python 3.9, а также установщик пакетов pip. При использовании системы уже был учтен график проверки ее стабильности, который включает проверку наличия необходимых ресурсов для создания чат-ботов. Все необходимые библиотеки уже включены в комплект поставки системы, и при ее установке любые недостающие пакеты будут установлены автоматически, чтобы обеспечить безупречную работу системы.

**Установка и настройка платформы**

Установите и настройте low-code платформу, которая будет использоваться для разработки чат-ботов.

Для проверки правильной установки платформы, можно осуществить следующие шаги:

1. Откройте конфигурационный файл.
2. Проверьте значения трех параметров расположенных в конце файла: <IsExistPython>, <IsExistEnvironment> и <IsExistLibrary>.
3. Убедитесь, что значения этих параметров установлены на "true".
4. Если значения соответствуют указанному, это свидетельствует о правильной установке платформы.

Пример конфигурационного файла системы Листинг 3.2.1 –

Конфигурационный файл системы.

<?xml version="1.0" encoding="utf-8"?>

<RuntimeSystem xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema">

<Environment>

<Path>C:\Users\TestEnv </Path>

<Name>.bot</Name>

</Environment>

<PythonInfo>

<IsInstalled>true</IsInstalled>

<Version>Python 3.9.13</Version>

<PipVersion>Name: pip Version: 22.2.2</PipVersion>

</PythonInfo>

<PythonLibraryInfo>

<IsInstalled>true</IsInstalled>

<NameLibrary>aiogram</NameLibrary>

<Version>2.23</Version>

</PythonLibraryInfo>

<IsExistPython>true</IsExistPython>

<IsExistEnvironment>true</IsExistEnvironment>

<IsExistLibrary>true</IsExistLibrary>

</RuntimeSystem>

**Разработка функциональности**

Используя визуальные инструменты и возможности low-code платформы, разработайте функциональность чат-ботов. Это может включать создание команд, обработку сообщений и другие требуемые функции.

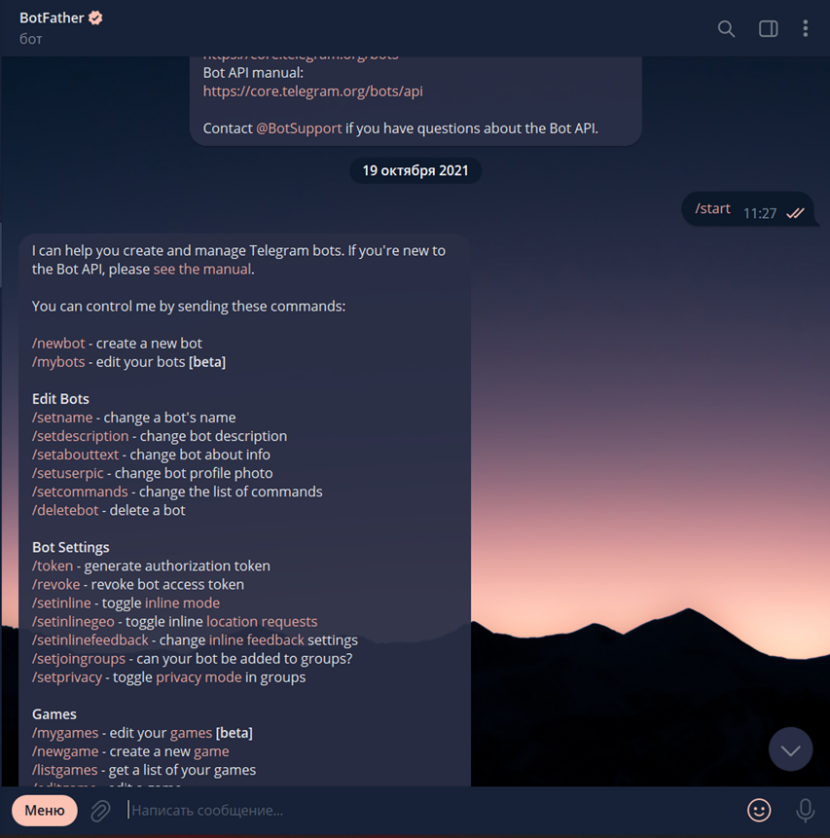
В качестве сервера для развертки бота, может выступить машина на которой установлена low-code платформа

3.3 Руководство пользователя

**Получение токена**

Для того чтобы получить токен для вашего бота в мессенджере Telegram, необходимо выполнить несколько шагов:

1. Откройте приложение Telegram и найдите в нем бота под названием BOT Father.
2. Если вы уже инициировали работу с ботом, переходите к следующему пункту. В противном случае, отправьте боту сообщение с текстом «/start», чтобы начать взаимодействие с ним (Рисунок 3.3.1).

  
Рисунок 3.3.1 – Начало работы с BOT Father

1. Чтобы создать нового бота, отправьте сообщение с текстом "/newbot" боту. Доступные команды можно просмотреть в меню (Рисунок 3.3.2). Если вы уже создали бота, перейдите к следующему пункту.

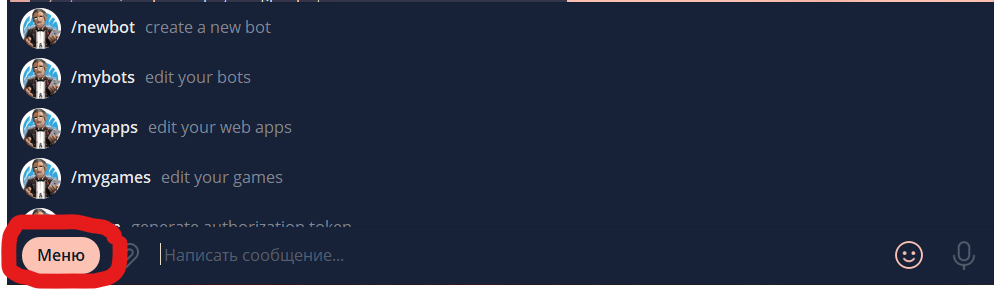
****

Рисунок 3.3.2 – Просмотр доступных команд

1. Дайте боту имя, которое будет отображаться в списке контактов, а затем придумайте уникальное имя для вашего бота (Рисунок 3.3.3), которое будет заканчиваться на слово «bot». Например, «test\_bot».

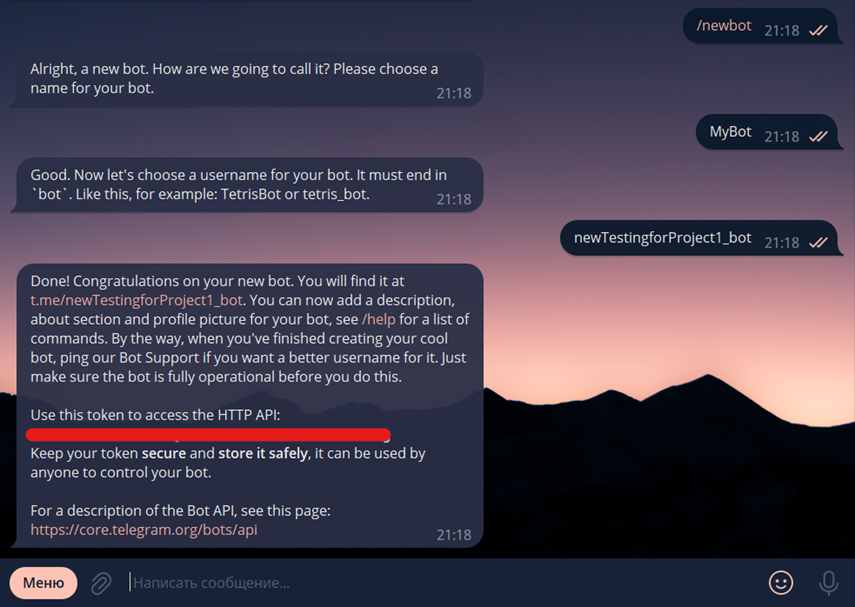


Рисунок 3.3.3 – Основные команды для получения токена

1. После этого BOT Father выдаст вам токен, который необходимо сохранить. Токен выглядит примерно так: 123456789:AbCdEfGhIjKlMnOpQrStUvWxYz.
2. Скопируйте токен и вставьте его в поле создания проекта, так как он потребуется системе для создания вашего бота.

**Меню проектов**

При запуске приложения открывается окно с меню проектов (Рисунок 3.3.4), где отображается список недавних проектов, а также имеются кнопки для создания нового проекта и открытия существующего.

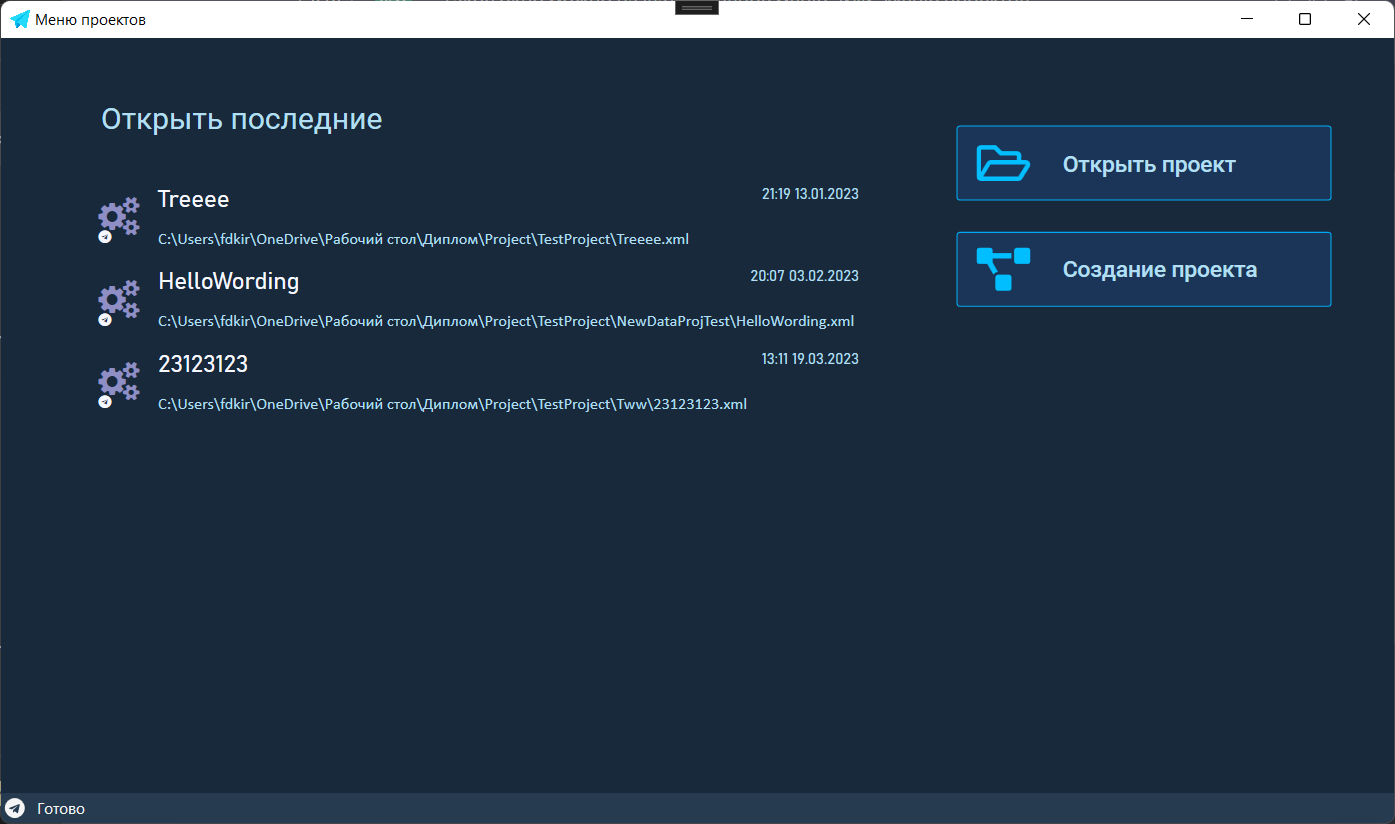


Рисунок 3.3.4 – Окно меню проектов

Возможны 4 варианта дальнейших действий:

1. Открыть нужный проект, дважды нажав на его название левой кнопкой мыши.
2. Нажать кнопку "Открыть проект" и выбрать файл проекта в диалоговом окне (Рисунок 3.3.5). Файл проекта должен соответствовать шаблону: Название\_проекта.xml. Если будет выбран некорректный файл система выведет сообщение об ошибке. Окно с сообщением об ошибке можно закрыть и попытаться выбрать другой файл.

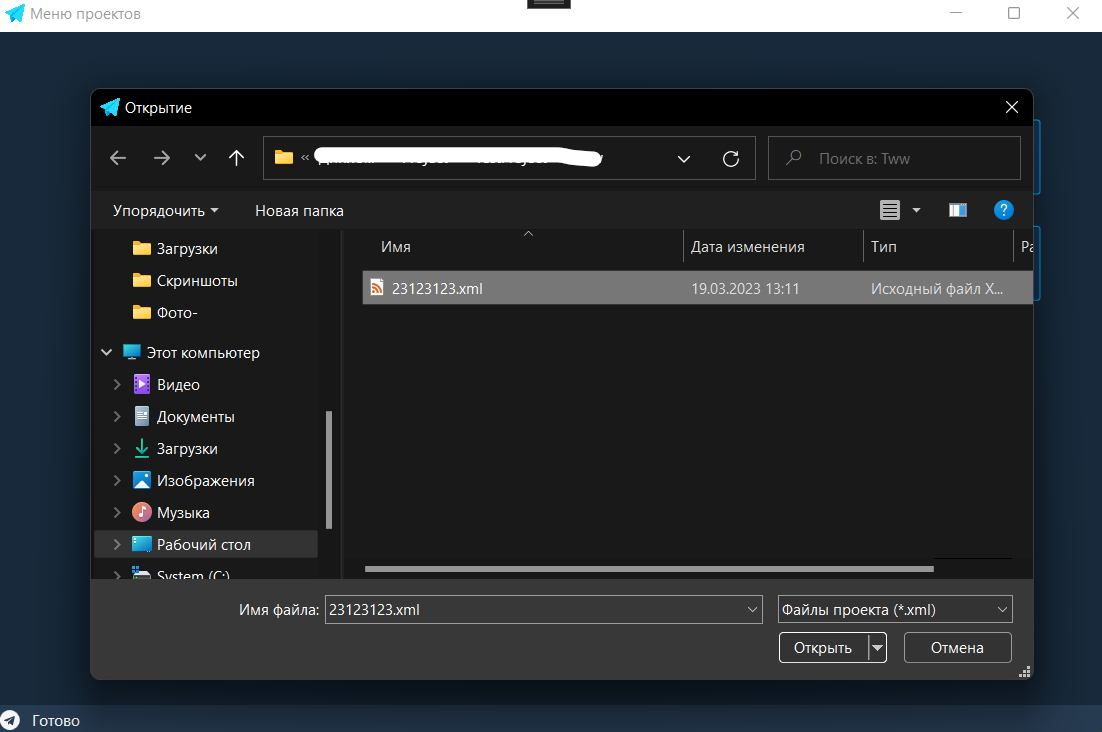


Рисунок 3.3.5 – Диалоговое окно выбора открываемого проекта

1. Создать новый проект, нажав на кнопку "Создание проекта" и заполнив необходимые поля в появившемся окне. Чтобы указать где будет располагаться проект можно нажать на кнопку «…» и тогда откроется диалоговое окно с выбором необходимой директории, или можно указать путь вручную в соответствующем поле.

Для продолжения работы с приложением, необходимо заполнить поле "Токен" обязательным символьным кодом, выданным вам BOT Father в мессенджере Telegram. Получить токен можно, ознакомившись с подробной инструкцией в разделе "Получение токена".

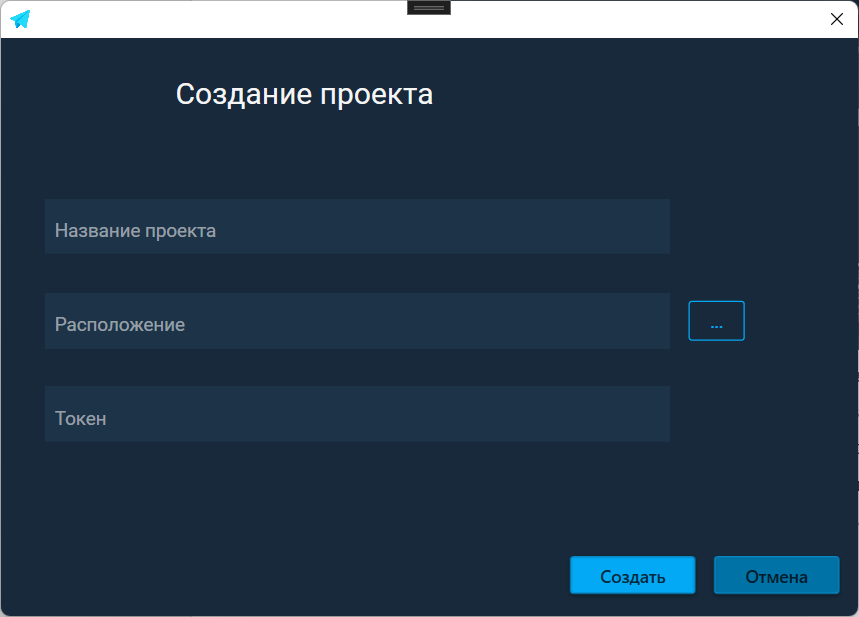


Рисунок 3.3.6 – Окно создания проекта

1. Удалить проект, нажав правой кнопкой мыши на названии проекта, который необходимо удалить, и выбрав опцию "удалить" (Рисунок 3.3.7).

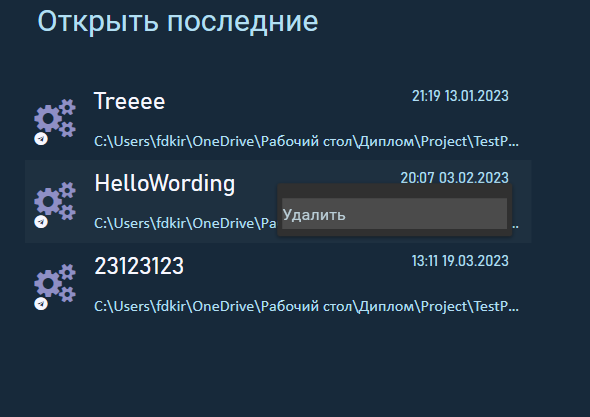


Рисунок 3.3.7 – Контекстное меню последних созданных проектов

**Конструктор чат-бота**

В данном руководстве представлено описание окна конструирования бота (Рисунок 3.3.8). Окно предоставляет удобный и интуитивно понятный интерфейс для создания поведения и управления ботом.

На верхней панели расположено меню, в котором пользователь может сохранить проект, открыть уже созданный проект или создать новый. Кнопка "Создать бота" позволяет создать нового бота и открыть окно управления ботом.

Центральная часть окна представляет собой древовидный список из команд и кнопок, которые пользователь может использовать для создания логики работы бота. Под древовидным списком находится то же самое представление в виде графа, что позволяет увидеть связи между различными элементами бота.

Справа от центральной части находится настройка команд, которая позволяет настроить параметры каждой команды и добавить необходимую логику для работы бота.

Пользовательский интерфейс панели элементов обеспечивает удобный доступ к различным элементам бота, таким как текстовые сообщения, картинки, файлы, кнопки и др.

Данный интерфейс предоставляет пользователю возможность создавать ботов с различными функциональными возможностями, управлять ими и настраивать их параметры для достижения нужных целей.

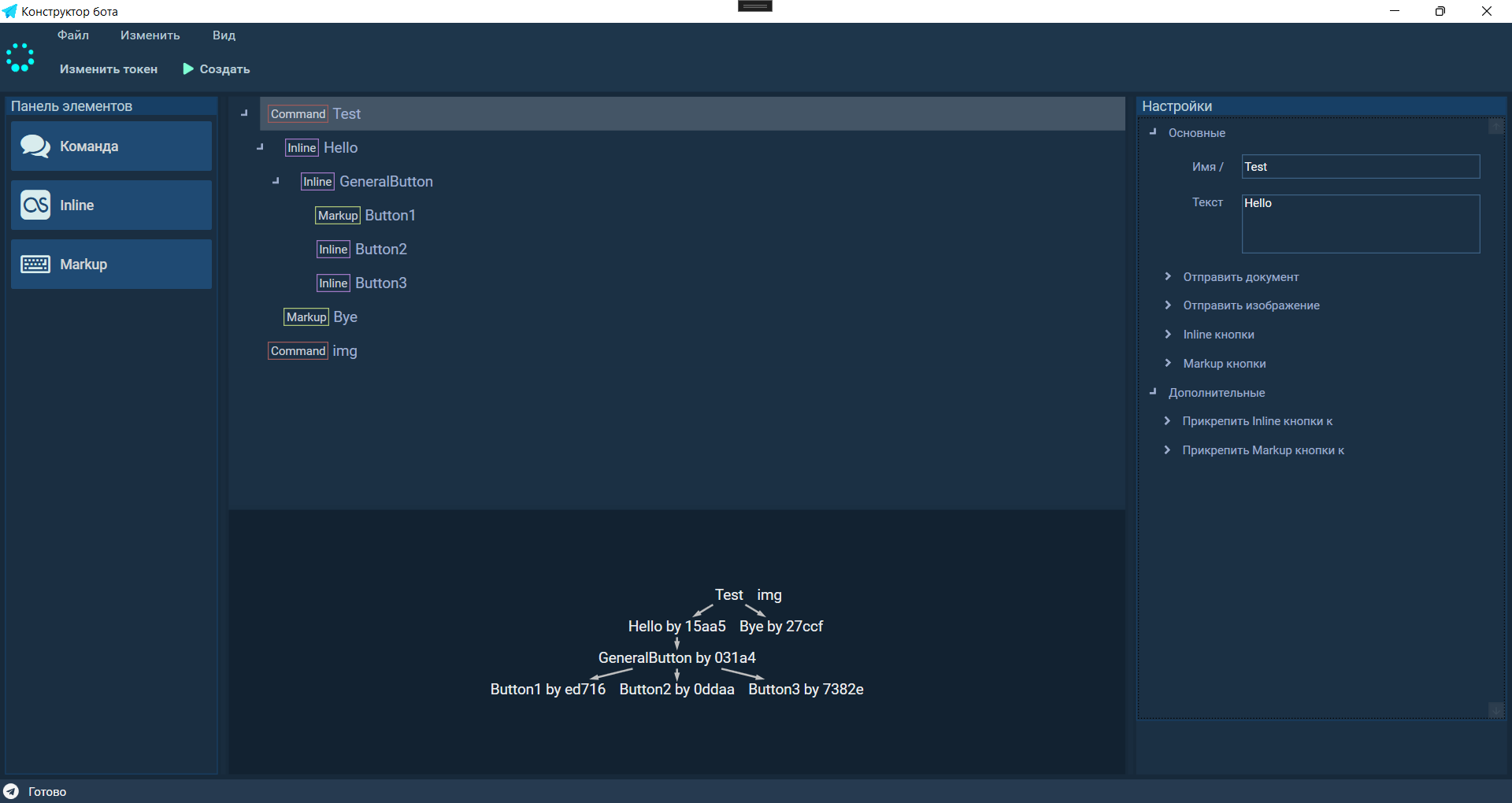


Рисунок 3.3.8 – Общий вид окна «Конструктор бота»

Меню представляет собой элемент управления, состоящий из 3 подменю «Файл», «Изменить» и «Вид»

Подменю «Файл»(Рисунок 3.3.9) содержит в себе кнопки:

1. Создать проект
2. Открыть проект
3. Сохранить проект
4. Выход

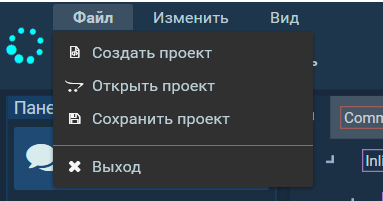


Рисунок 3.3.9 – Меню «Файл»

Подменю «Изменить» (Рисунок 3.3.10) содержит в себе кнопку «Изменить токен». Она предназначена для изменения ключа который выдал BOT Father

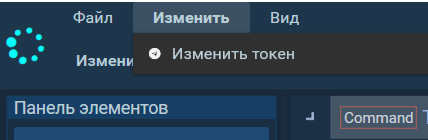


Рисунок 3.3.10 – Меню «Изменить»

В подменю «Вид» (Рисунок 3.3.11) находится кнопка «Убрать график» или «Показать график». В зависимости от текущего отображения графика, эта кнопка изменяет его видимость.

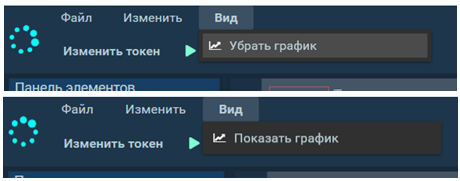


Рисунок 3.3.11 – Меню «Вид»

Настройки команд и кнопок (Рисунок 3.3.12) включают в себя два основных параметра: «имя» команды и текст. Пункт «текст» необязателен, тогда как пункт «имя» обязателен для каждой команды и кнопки. Если вы создаете Markup кнопку, то имя должно быть уникальным, так как обработка сообщений происходит по имени кнопки. Если вы введете имя, которое уже существует, то система сообщит об этом.

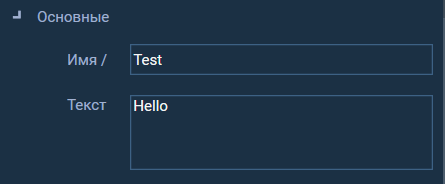


Рисунок 3.3.12 – Основные параметры команд и кнопок

Отправка документов и изображений осуществляется через соответствующие разделы (Рисунок 3.3.13). Каждый раздел имеет поле для ввода пути к файлу на локальном компьютере и поле для подписи файла. Для добавления файлов необходимо нажать кнопку "Добавить файлы", после чего откроется соответствующее окно (Рисунок 4.3.14).

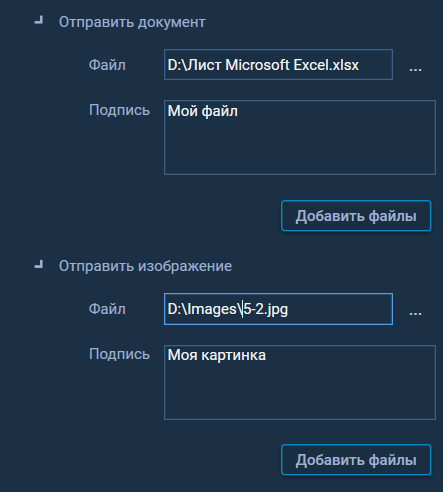


Рисунок 3.3.13 – Настройки отправки документов и изображений

Окно добавления файлов (Рисунок 3.3.14), реализованном аналогично настройкам в окне «Конструктор бота», в нем может добавить несколько файлов. При нажатии на кнопку «ОК» добавленные файлы будут сохранены, а при нажатии на кнопку «Отмена» добавление файлов будет отменено.

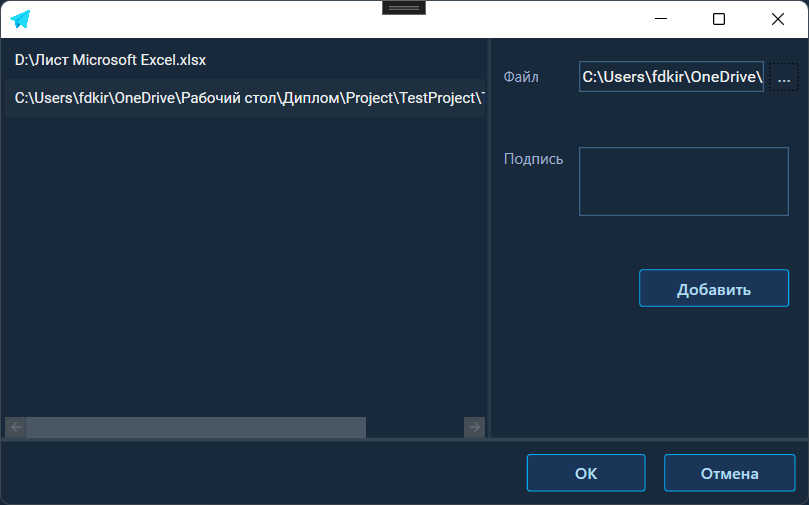


Рисунок 3.3.14 – Окно добавления нескольких файлов

В разделе Inline-кнопок пользователь может добавлять кнопки, указывая их название и ссылку, которую они должны открывать при нажатии. Кнопка "Добавить" добавляет новую кнопку в список и сразу же отображает ее в дереве команд. В поле "Количество кнопок" пользователь указывает число кнопок, которые должны располагаться в строке под сообщением. Далее пользователь выбирает, к чему прикрепить кнопки из предоставленных опций. В подразделе "Кнопки" отображаются все дочерние кнопки.

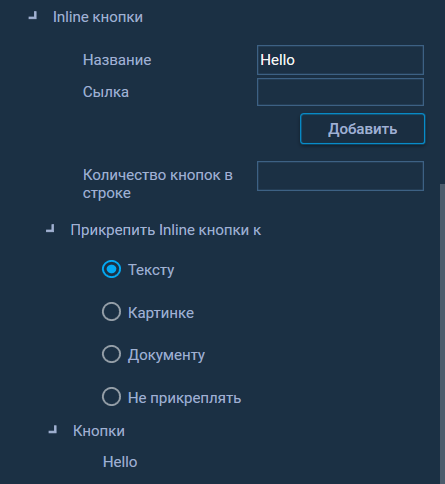
****

Рисунок 3.3.15 – Настройки добавляемой inline кнопки

Markup-кнопки имеют аналогичный раздел с Inline-кнопками, за исключением того, что название кнопки должно быть уникальным, так как идентификация кнопки происходит по ее названию.

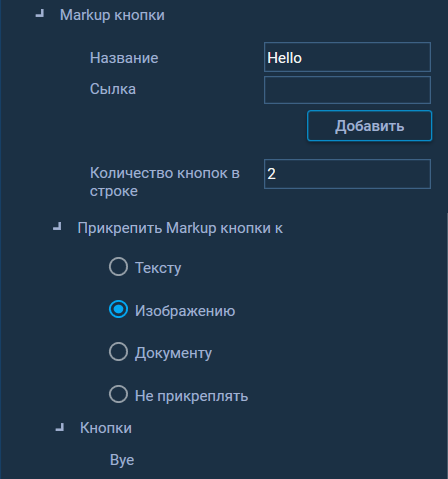
****

Рисунок 3.3.16 – Настройки добавляемой markup кнопки

Окно управления чат-ботом (Рисунок 3.3.17) позволяет следить за состоянием работы бота. Здесь находится список событий и время каждого события. В данном окне можно остановить или запустить работу бота. При открытии окна бот автоматически запускается.

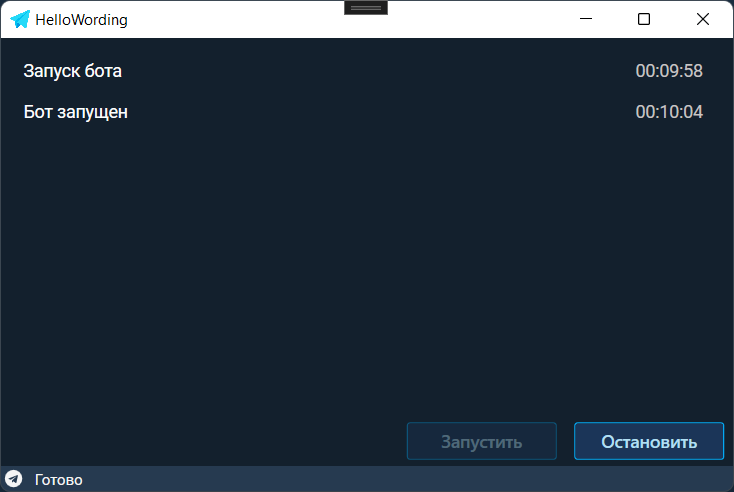


Рисунок 3.3.17 – Окно состояния бота

3.4 Экономическая часть

При реализации проекта необходимо определить целесообразность данной разработки и рассчитать экономическую эффективность данного проекта.

В первую очередь определим трудоемкость разработки.

Трудоемкость разработки представлена в таблице 3.4.1. Строка «Всего» отображает общую трудоемкость разработки.

Таблица 3.4.1 – Трудоемкость разработки программного продукта

|  |  |
| --- | --- |
| **Наименование этапа** | **Трудоемкость выполнения этапа, час** |
| Анализ low-code платформ для создания чат-ботов | 48 |
| Проектирование архитектуры приложения | 120 |
| Разработка функциональности | 175 |
| Тестирование приложения | 70 |
| Всего | 413 |

Для реализации проекта возникает потребность в наемен сотрудников. Для наибольшей оптимизации процесса разработки и дальнейшем обеспечении работоспособности платформы необходимо нанять программиста-разработчика, который будет заниматься разработкой платформы и руководителя проекта, который будет контролировать все процессы работы.

Исполнителем является junior .Net программист. Его оклад по Калужской области составляет в среднем 75000 рублей в месяц [11]. Потребность в количестве персонала и заработная плата сотрудников отображены в таблице 3.4.2.

Таблица 3.4.2 – Затраты на оплату труда

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнители | Ставка, руб/час | Количество сотрудников | Оплата труда за весь срок работы, руб. |
| Программист-разработчик | 426 | 1 | 175938 |
| Руководитель проекта | 568 | 1 | 234584 |
|  | | | 410522 |

Дополнительная заработная плата составляет 12 % от основной заработной платы. Дополнительная заработная плата за весь период разработки составит:

З доп,зп = 410522\*0,12 = 49263 руб.

Налоги на заработную плату составляют 30 % от суммы основной и дополнительной заработной платы:

З н = (410522+49263) \*0,3= 137935 руб.

Также для реализации данного проекта возникает необходимости в приобретении технических средств: ноутбуки для сотрудников в количестве 2-х штук.

В таблице 3.4.3 отображении необходимые технические средства и их стоимость.

Таблица 3.4.3 – Затраты на технические средства

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование ТС** | **Цена, руб.** | **Количество, шт.** | **Стоимость, руб.** |
| Ноутбук LENOVO | 35000 | 2 | 70000 |
| Итого: | | | 70000 |

Кроме того, в процессе реализации проекта будут возникать расходы на электричество и интернет для сотрудников. Работа сотрудников будет осуществляться удаленно, но работодателю необходимо компенсировать данные расходы.

В таблице 3.4.4 представлены возникающие затраты на расходные материалы.

Таблица 3.4.4 – Затраты на расходные материалы

| **Затраты** | **Стоимость** | **Количество** | **Сумма, руб.** |
| --- | --- | --- | --- |
| Интернет | 1500 руб./месяц | 4 месяца | 6000 |
| Электричество | 6 руб./КВт\*ч | 800 КВт\*ч | 4800 |
| Итого: | | | 10800 |

После окончания разработки платформы необходимо привлечь потенциальных клиентов. В результате чего возникнут расходы на маркетинг и продвижение, которые составят 120.000 руб.

Общие затраты за весь период разработки системы составят:

Зоб = З осн,зп + З доп,зп+ З н + Зтс + Зрм + Зм (1)

410522 + 49263 +137935 + 70000 + 10800 + 120000 = 895720 руб.

Вторым этапом необходимо рассчитать потенциальные доход, который будет получен после реализации проекта. Для этого необходимо рассчитать число потенциальных клиентов. От всего объема среднего и малого бизнеса, компании, которые заинтересованы в развитии своего бизнеса в соцсетях и мессенджерах составляет около 5%, а это приблизительно 300 тыс. компаний.

За первый год планируется охватить 1% от объема потенциальных клиентов, что составит 3 тыс. компаний, из которых 20% приобретут low-code платформу, а это 600 компаний.

Рассчитаем прибыль за первый год реализации, если стоимость платформы составит 1.000 руб.:

W = P\*Q, где (2)

W – выручка,

P – цена за одну единицу товара

Q – количество реализованной продукции

W = 600\*1.000 = 600.000 руб.

Для нахождения срока окупаемости, применим следующую формулу:

, (3)

где РР - срок окупаемости, выраженный в годах;

– сумма вложенных средств;

W – выручка от реализации за год.

Таким образом low-code платформа для создания чат-ботов в мессенджере Telegram может окупиться за 18 месяцев.

Так можно сделать вывод, что разработка данного проекта целесообразна и в будущем, разработка платформы окупится и проект будет приносить прибыль.

3.5 Эргономика проектного решения

Эргономика занимается изучением и проектированием трудовой деятельности с целью оптимизации условий труда и повышения профессионального мастерства. Она фокусируется на системе "человек - орудие труда - предмет труда - производственная среда". Эргономика имеет значительное значение при организации рабочего места и учитывает антропометрические, физические и психологические требования.

В случае рабочего места инженера по информационной безопасности, следует учитывать следующие основные условия:

Площадь на одно рабочее место пользователя ПЭВМ с ЭЛТ должна быть не менее 6 м2.

Высота рабочей поверхности стола для взрослых пользователей должна регулироваться в диапазоне 680 - 800 мм.

Размеры рабочей поверхности стола для ПЭВМ должны быть модульными, например, ширина 800, 1000, 1200 и 1400 мм, глубина 800 и 1000 мм при фиксированной высоте 725 мм.

Рабочий стол должен иметь определенные размеры, например, высоту 600 мм, ширину 500 мм, глубину 450 мм на уровне согнутых ног и 650 мм на уровне вытянутых ног.

Конструкция рабочего стула должна предусматривать регулировку высоты сиденья (400 - 550 мм) и угла наклона, а также регулировку подлокотников по высоте и расстоянию между ними.

Рабочее место пользователя ПЭВМ должно быть оборудовано подставкой для ног, которая имеет ширину не менее 300 мм, глубину не менее 400 мм, регулировку по высоте в пределах до 150 мм и по углу наклона опорной поверхности подставки до 20°. Рекомендации по расположению за компьютером показаны на рисунке 3.5.1

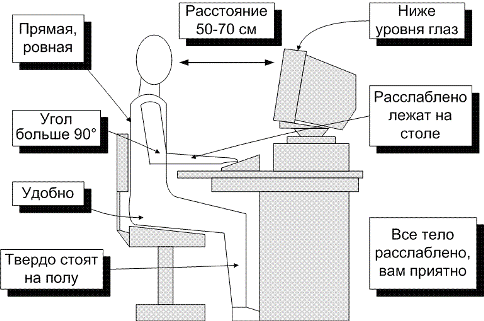


Рисунок 3.5.1 – Рекомендации по расположению за компьютером

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы было разработано приложение, которое позволяет создавать чат-ботов для мессенджера Telegram с использованием low-code платформы.

В рамках работы были разработаны модуль создания ботов, модуль отладки и модуль взаимодействия с данными. Были учтены основные сценарии использования платформы и предусмотрены настройки для создания чат-ботов.

Окончательное тестирование приложения позволило выявить и исправить возможные ошибки и недочеты. Была проверена работоспособность платформы, а также корректность формирования бота, чтобы обеспечить стабильную и безопасную работу приложения.

В дальнейшем развитии проекта можно рассмотреть возможность добавления новых функций и интеграцию с другими популярными мессенджерами, чтобы расширить аудиторию пользователей и увеличить функциональность платформы.

В процессе выполнения работы я углубил свои знания в предметной области, провел анализ аналогов и прототипов, а также осуществил выбор подходящих инструментов и платформы для разработки.

В результате данной работы были успешно решены все поставленные задачи, и приложение готово к использованию.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Material Design In XAML [Электронный ресурс] – URL: <http://materialdesigninxaml.net/> дата посещения: 19.02.2023
2. Telegram API. [Электронный ресурс] – URL: https://core.telegram.org/ дата посещения: 19.01.2023
3. Дадонов, П. Е. Особенности реализации паттерна Observer средствами языка программирования C# / П. Е. Дадонов, О. А. Молокович // Молодежный вестник Уфимского государственного авиационного технического университета. – 2020. – № 1(22). – С. 56-60. – EDN JVRYJK.
4. Документация Aiogram. [Электронный ресурс] – URL: <https://docs.aiogram.dev/en/latest/> дата посещения: 11.12.2022
5. Документация NLog [Электронный ресурс] – URL: https://github.com/nlog/nlog/wiki дата посещения: 12.02.2023
6. Месарош, Д. Шаблоны тестирования xUnit : рефакторинг кода тестов / Д. Месарош ; Джерард Месарош ; [пер. с англ. и ред. О. А. Лещинского ; предисл. Мартина Фаулера]. – Москва [и др.] : Вильямс, 2009. – 831 с. – ISBN 978-5-8459-1448-4. – EDN QMTBPV.
7. Многопоточность [Электронный ресурс] – URL: <https://metanit.com/sharp/tutorial/11.1.php> дата посещения: 21.02.2023
8. Оптимизация приложений на платформе .NET с использованием языка C#. ДМК Пресс. Дима Зурбалев, Идо Флатов, Саша Голдштейн, 526 стр ISBN:978-5-94074-944-8
9. Петрик, Е. А. Проектирование программного интерфейса на основе опыта пользователей / Е. А. Петрик, Д. А. Фомин, И. С. Архипова // Программная инженерия: современные тенденции развития и применения (пи-2020) : сборник материалов IV Всероссийской научно-практической конференции, посвященной 30-летию создания кафедры программной инженерии, Курск, 12–13 марта 2020 года. – Курск: Юго-Западный государственный университет, 2020. – С. 77-82. – EDN DGKXCM.
10. Привязка данных с помощью WPF [Электронный ресурс] – URL: <https://docs.microsoft.com/ru-ru/ef/ef6/fundamentals/databinding/wpf> дата посещения: 03.03.2023
11. Радио sputnik Зарплата программиста. [Электронный ресурс] – URL: <https://radiosputnik.ria.ru/20220129/programmist-1769412346.html> дата посещения: 25.04.2023
12. Хорошхин, В. К. Использование паттерна Singleton в языках программирования С++, С#, Java / В. К. Хорошхин // Труды университета. – 2011. – № 1(42). – С. 82-84. – EDN TFDMKF.
13. Шаблон Model-View-ViewModel [Электронный ресурс] – URL: <https://docs.microsoft.com/ru-ru/xamarin/xamarin-forms/enterprise-application-patterns/mvvm> дата посещения: 01.12.2022
14. Шилов, М. В. Параллельный алгоритм обхода всех путей бинарного дерева / М. В. Шилов // Вестник Ростовского государственного университета путей сообщения. – 2008. – № 1(29). – С. 54-60. – EDN JPJCRR.

ПРИЛОЖЕНИЕ А

*UML диаграмма прецедентов*

