**Содержание**

[Введение 2](#_Toc136562851)

[1. Анализ предметной области и выбор технологий для разработки 3](#_Toc136562852)

[1.1. Анализ предметной области 3](#_Toc136562853)

[1.2. Выбор технологий для разработки 4](#_Toc136562854)

[2. Архитектура системы 6](#_Toc136562855)

[3. Проектирование базы данных 7](#_Toc136562856)

[4. Разработка клиентской составляющей веб-приложения 10](#_Toc136562857)

[4.1 Структура React приложения 11](#_Toc136562858)

[4.2 Компоненты React приложения 12](#_Toc136562859)

[5. Руководство пользователя 14](#_Toc136562860)

[5.1. Авторизация 14](#_Toc136562861)

[5.2. Список ботов 15](#_Toc136562862)

[5.3. Конструктор 17](#_Toc136562863)

[5.4. Проверка работы бота 21](#_Toc136562864)

[6. Перспективы 24](#_Toc136562865)

[Заключение 25](#_Toc136562866)

[Список использованных источников 26](#_Toc136562867)

# Введение

Чат-боты становятся все более популярными и широко используемыми в различных сферах, включая бизнес, образование, здравоохранение и многие другие. Использование чат-ботов позволяет автоматизировать процессы, улучшить взаимодействие с клиентами и повысить эффективность работы. С увеличением спроса на чат-боты возрастает и потребность в инструментах, таких как "Конструктор чат-ботов", которые облегчают и ускоряют процесс создания и настройки чат-ботов.

Они помогают имитировать диалог с пользователем путём привязки команд к ключевым фразам. Это помогает избавится от участия человека в информировании пользователей.

Их использование хорошо подходит большим компаниям для автоматизированного ответа на частые вопросы клиентов и сотрудников, отправки файлов и рассылки новостей. Также чат-боты используются для получения различной направленной информации (например, поисковик по базе фильмов).

Создание ботов для мессенджеров является настолько не трудной задачей, что любой человек изучив пару обучающих видео сможет написать его.

На этом этапе проблем не возникает, но увеличивая структуру запросов и создавая новых ботов становится всё труднее их поддерживать, так как даже для программиста возвращение к старому коду создаёт много вопросов и приходится заново изучать то, что было написано ранее. Вывод: такой подход занимает много времени.

Также существуют сайты предлагающие услуги конструктора и поддержки ботов, но конечно это не бесплатно.

В компании было принято решение разработать своё ПО для этих нужд.

В курсовой работе будет рассмотрена разработка клиентской составляющей веб-приложения, а также проектирование базы данных.

## Анализ предметной области и выбор технологий для разработки

Веб-приложение "Конструктор чат-ботов" предназначено для создания и настройки чат-ботов для различных платформ, в данном случае реализуется создание чат-ботов для платформы Telegram. Чат-боты позволяют автоматизировать процесс общения с пользователями, предоставляя информацию, отвечая на вопросы, выполняя задачи и предоставляя интерактивные сценарии.

### Анализ предметной области

При анализе предметной области веб-приложения "Конструктор чат-ботов" следует учитывать следующие аспекты:

**Функциональные требования:**

* **Создание ботов**: Пользователь должен иметь возможность создавать новые чат-боты для платформы Telegram.
* **Настройка ботов**: Пользователь должен иметь возможность настраивать параметры ботов, такие как имя, аватар, приветственное сообщение и команды.
* **Управление командами**: Пользователь должен иметь возможность настраивать команды, определять текстовые ответы или скрипты, которые будут выполняться при получении определенных команд от пользователя.
* **Создание сценариев**: Пользователь должен иметь возможность создавать интерактивные сценарии, определяющие последовательность вопросов и ответов для взаимодействия с пользователем.
* **Интеграция с внешними сервисами**: Пользователь должен иметь возможность интегрировать чат-ботов с различными внешними сервисами и API для получения и отправки данных.
* **Аналитика использования**: Пользователь должен иметь доступ к статистике использования чат-ботов, такой как количество подписчиков, количество запросов, популярные команды.

**Нефункциональные требования:**

* **Производительность**: Приложение должно обеспечивать высокую производительность для быстрого создания и настройки чат-ботов, а также эффективного обработки запросов от пользователей.
* **Масштабируемость**: Приложение должно быть масштабируемым, чтобы легко справляться с ростом числа пользователей и созданных чат-ботов.
* **Безопасность**: Приложение должно обеспечивать безопасность данных пользователей и настроек чат-ботов, а также предотвращать несанкционированный доступ к функциональности и административным возможностям.
* **Доступность**: Приложение должно быть доступным для пользователей на различных устройствах и с разными ограничениями, такими как доступность для людей с ограниченными возможностями.
* **Удобство использования интерфейса**: Приложение должно иметь интуитивно понятный и удобный пользовательский интерфейс, который позволяет легко создавать, настраивать и управлять чат-ботами.

**Интеграция с Telegram API:**

* **Интеграция с Telegram**: Приложение должно взаимодействовать с Telegram API для создания и настройки ботов, отправки и получения сообщений, управления подписчиками и других функций, предоставляемых Telegram API.

**Управление данными:**

* **Хранение данных**: Приложение должно предоставлять механизм для хранения данных о созданных ботах, настройках, командах, сценариях и подписчиках.
* **Безопасность данных**: Данные, хранящиеся в приложении, должны быть защищены с использованием соответствующих мер безопасности, таких как шифрование и аутентификация.

В процессе разработки веб-приложения необходимо уделить внимание требованиям пользователей, проанализировать возможности и ограничения Telegram API, выбрать подходящие технологии и инструменты разработки, а также рассмотреть вопросы безопасности и масштабируемости приложения.

### Выбор технологий для разработки

1. **React JS:**

React JS является популярной и мощной библиотекой для разработки пользовательского интерфейса. Она обеспечивает быструю и эффективную отрисовку компонентов, удобное управление состоянием приложения и простоту переиспользования кода. Благодаря виртуальному DOM, React JS обеспечивает быстрое обновление только измененных частей интерфейса.

1. **Axios:**

Axios является удобной библиотекой для выполнения HTTP-запросов в браузере или на сервере. В контексте "Конструктора чат-ботов", Axios позволяет установить связь между клиентской и серверной частями приложения. Он обеспечивает возможность отправки и получения данных.

1. **Django:**

Django является мощным фреймворком для разработки веб-приложений на языке Python. Он обеспечивает высокую производительность, безопасность и масштабируемость. В контексте "Конструктора чат-ботов", Django может быть использован для взаимодействия с базой данных и управления бизнес-логикой приложения.

1. **MSSQL:**

Выбор конкретной системы управления базами данных (СУБД) зависит от требований проекта, но MSSQL (Microsoft SQL Server) является одной из популярных и широко используемых СУБД. Она предоставляет надежное хранение данных, обеспечивает высокую производительность и масштабируемость. MSSQL может быть использована для хранения данных о пользователях, созданных чат-ботах, настройках, командах, сценариях и подписчиках

1. **C#:**

C# является мощным языком программирования, который может быть использован для разработки программной логики, поддерживающей работу чат-ботов. Он имеет богатую экосистему инструментов и библиотек для работы с API, обработки сообщений, выполнения бизнес-логики и интеграции с другими сервисами. Использование C# позволяет разрабатывать функциональность чат-ботов, такую как обработка команд, генерация ответов и взаимодействие с внешними сервисами.

Указанные технологии (React JS, Axios, Django, MSSQL и C#) были выбраны с учетом их преимуществ, популярности, поддержки сообществом разработчиков, соответствия требованиям проекта, знаний и опыта команды. Также они хорошо подходили под технологии, использующиеся в компании, для которой и реализовывался конструктор.

## Архитектура системы

Архитектура системы представлена на рисунке 2.1.

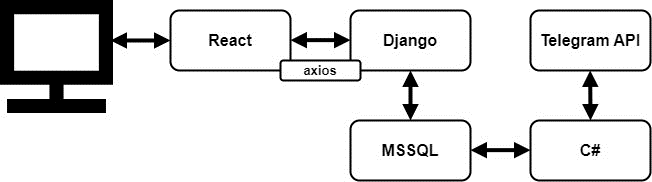


Рисунок 2.1 – Архитектура системы

Пользователь контактирует с веб приложением через клиентскую часть, написанную на React JS. Она в свою очередь взаимодействует с серверной частью через запросы, используя axios. Django реализует запросы для получения информации из базы данных MSSQL.

Программа, осуществляющая поддержку ботов, написана на C#. Она анализирует базу данных и поддерживает подключённых ботов связываясь с Telegram API используя специализированную библиотеку.

Программа на C# рассматривается, в данной курсовой работе, как отдельная самостоятельная часть проекта и расписана не будет.

Плюсы разделения клиентской и серверной частей на два веб-сервера:

1. **Расширяемость**: Эта архитектура позволяет разработчикам легко расширять функциональность как на клиентской, так и на серверной стороне. Это может быть полезно при введении новых функций и возможностей.
2. **Улучшенная производительность**: Разделение клиентской и серверной стороны на два веб-сервера может дать увеличение производительности, так как браузер и сервер могут параллельно отправлять запросы и обрабатывать ответы. Кроме того, серверная часть может лучше обрабатывать более сложные задачи.
3. **Легкость развертывания**: Разведение клиентской и серверной стороны может упростить процесс развертывания и управления обновлениями на каждой стороне, что может быть особенно полезно в больших проектах с большим количеством разработчиков.
4. **Хорошая распределённость нагрузки**: Разделение клиентской и серверной стороны на два сервера может помочь в распределении нагрузки, что позволит лучше масштабировать приложение в случае увеличения трафика.

Далее в курсовой рассматриваются только проектирования базы данных и реализация фронт части веб-приложения, которая была реализована в результате работы над совместным проектом.

## Проектирование базы данных

Для регистрации пользователя необходимы его электронная почта, логин и пароль. Также к пользователю привязывается его идентификатор.

Для регистрации бота необходимы его идентификатор пользователя, имя, уникальное имя (идентификатор бота в Telegram) и токен подключения к боту. Также у бота есть поле отвечающие за статус бота (включён/выключен).

Должен вестись список подключённых пользователей к чат-ботам.

Для добавления команд боту необходима следующая информация: идентификатор бота, имя команды и тип команды. Тип команды берётся из справочника команд, в нём хранится идентификатор и название типа.

К каждой команде должна быть привязана запись из таблицы (сообщение/рассылка), соответствующей типу команды.

Для добавления сообщения необходимы идентификатор команды и сообщение.

Для добавления рассылки необходимы идентификатор команды, сообщение, дата и время рассылки.

Также к команде может быть привязана запись из таблицы «Медиа», в которой хранится идентификатор команды, имя файла, тип файла и сам файл в формате base64.

У команды может быть фразы вызова, которые заносятся в соответствующую таблицу.

Команды могут быть связаны с другими командами. Такие связи заносятся в таблицу связей следующим образом: идентификатор текущей команды и идентификатор следующей команды.

Структура базы данных изображена на рисунке 3.

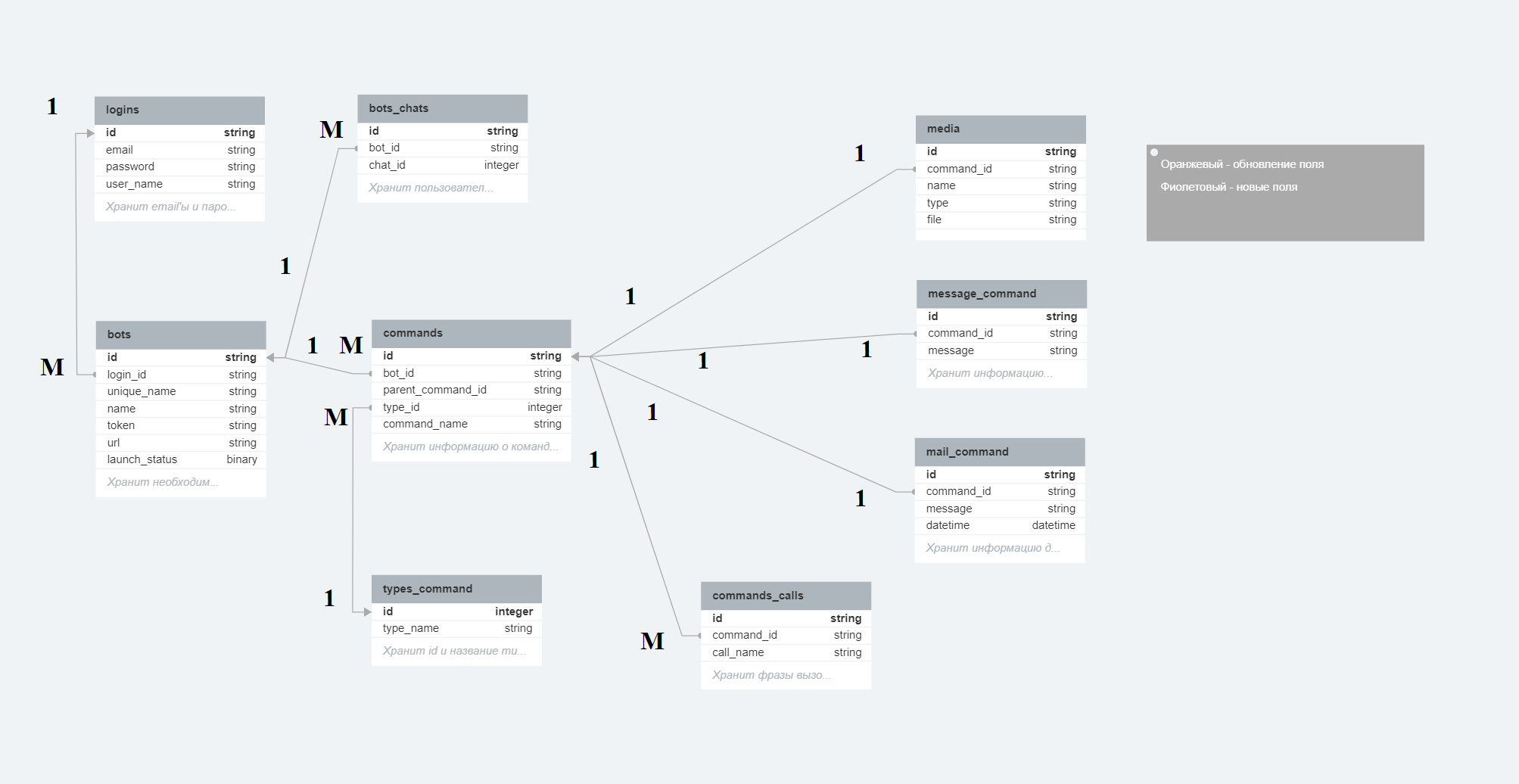


Рисунок 3.1 – Структура базы данных

Ниже приведен перевод полей таблиц:

Logins – регистрации

id – идентификатор;

email – электронная почта;

password – пароль;

user\_name – имя пользователя.

Bots – Боты

id – идентификатор;

login\_id – идентификатор пользователя;

unique\_name – имя-идентификатор бота;

name – имя бота;

token – ключ подключения бота;

url – адрес подключаемого бота;

launch\_status – состояние запуска.

Bots\_chats – чаты с ботами

id – идентификатор;

bot\_id – идентификатор бота;

chat\_id – идентификатор чата.

Commands – Боты

id – идентификатор;

bot\_id – идентификатор бота;

type\_id – идентификатор типа команды;

command\_name – название команды;

link\_status – статус связи команды.

Types\_command – типы команд

id – идентификатор;

type\_name – название типа.

Commands\_calls – вызовы команд

id – идентификатор;

command\_id – идентификатор команды;

call\_name – фраза вызова.

Link\_commands – связи команд

id – идентификатор;

current\_command – идентификатор текущей команды;

following\_command – идентификатор следующей команды.

Message\_command – команды сообщения

id – идентификатор;

command\_id – идентификатор команды;

message – сообщение.

Mail\_command – регистрации

id – идентификатор;

command\_id – идентификатор команды;

message – сообщение;

datetime – дата и время.

Commands – Боты

id – идентификатор;

command\_id – идентификатор команды;

name – название файла;

type – тип файла;

file – файл.

## Разработка клиентской составляющей веб-приложения

Клиентская часть веб-приложения на React JS является фронтенд-частью приложения, которая отвечает за пользовательский интерфейс и взаимодействие пользователя с приложением.

React JS - это JavaScript-библиотека, которая позволяет создавать пользовательские интерфейсы с использованием компонентного подхода. Она позволяет разделить пользовательский интерфейс на множество маленьких, независимых компонентов, которые могут быть повторно использованы и легко модифицированы.

Для начала разработки клиентской части приложения на React JS необходимо настроить окружение, установить необходимые зависимости и настроить сборку проекта. После этого можно приступить к созданию компонентов, которые будут составлять интерфейс приложения.

Компоненты в React JS могут быть функциональными или классовыми. В функциональных компонентах используется функция, возвращающая JSX (расширение синтаксиса JavaScript, которое позволяет описывать структуру пользовательского интерфейса). В классовых компонентах используется класс, который наследуется от базового класса React.Component и имеет метод render(), в котором возвращается JSX.

Компоненты могут содержать внутри себя другие компоненты, что позволяет создавать иерархию компонентов и компонентный подход позволяет удобно управлять состоянием приложения и обновлять интерфейс при изменении данных.

Клиентская часть приложения на React JS может включать в себя следующие основные элементы:

1. **Роутинг**: Можно использовать библиотеку React Router для настройки маршрутизации в приложении. Это позволяет определить различные пути URL и связать их с соответствующими компонентами, чтобы реализовать навигацию между разными страницами.
2. **Компоненты форм**: Для работы с формами можно использовать различные компоненты, такие как <input>, <select>, <textarea> и другие. Компоненты форм могут иметь своё собственное состояние и обрабатывать пользовательский ввод.
3. **Компоненты списка**: Если требуется отображение списка элементов, можно использовать компоненты списка, такие как <ul> и <li>. Компонент списка может принимать данные из внешнего источника, например, из базы данных или API, и отображать элементы списка на основе этих данных.
4. **Взаимодействие с API**: Чтобы получать и отправлять данные на сервер, можно использовать сетевые запросы с помощью библиотеки Axios или встроенного API Fetch. Это позволяет обновлять данные в приложении, обмениваться информацией с сервером и поддерживать актуальность данных.
5. **Стилизация**: Для стилизации компонентов можно использовать CSS или CSS-препроцессоры, такие как SASS или LESS. Также можно использовать библиотеки стилей, такие как Bootstrap или Material-UI, которые предоставляют готовые компоненты и стили для быстрой разработки интерфейса.
6. **Обработка событий**: В React JS можно добавлять обработчики событий к компонентам, например, для обработки кликов, изменения состояния или отправки данных на сервер. Это позволяет реагировать на действия пользователя и обновлять интерфейс соответствующим образом.

Компоненты, используемые на странице, могут быть повторно использованы в других частях приложения. Это позволяет создавать модульную и масштабируемую клиентскую часть. Важно обеспечивать низкую связность между компонентами, чтобы они были независимы друг от друга. Это повышает возможность расширения функциональности и надежности всего приложения.

В данной реализации большинство компонентов написаны с использованием классов.

Клиентская часть веб-приложения на React JS предоставляет пользователю удобный и отзывчивый интерфейс, который взаимодействует с серверной частью приложения для обработки данных и выполнения задач.

### 4.1 Структура React приложения

На рисунке 4.1.1 представлена иерархическая структура вызовов компонентов в React приложении.

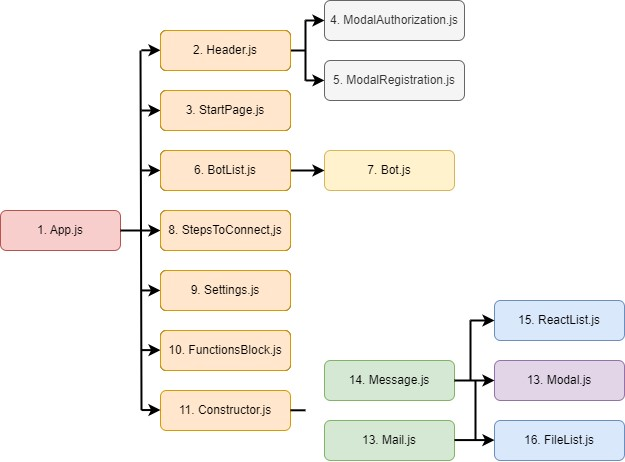


Рисунок 4.1.1 – Структура React приложения

### 4.2 Компоненты React приложения

* 1. App.js – в нём хранятся идентифицирующие данные пользователя, боты и вспомогательные данные для инициализации действий перемещения между страницами. Здесь же реализованы почти все функции отправки запросов на сервер, такие как: получение ботов, заполнения ботов, автоматической авторизации, выхода из аккаунта. Рендер осуществляет обновления и переход по страницам.
  2. Header.js – это компонент шапки сайта. В нём реализуются возврат к предыдущей странице и выход из аккаунта.
  3. StartPage.js – это компонент стартовой страницы для не авторизированного пользователя.
  4. ModalAuthorization.js – это компонент модального окна авторизации пользователя. В нём осуществляется отправка запроса на авторизацию пользователя.
  5. ModalRegistration.js – это компонент модального окна регистрации пользователя. В нём осуществляется отправка запроса на регистрацию пользователя.
  6. BotList.js – это компонент для списка ботов.
  7. Bot.js – это компонент отображения бота в списке ботов. В нём реализуются переход в конструктор бота для его изменения, отключение / подключение бота и удаления.
  8. StepsToConnect.js – это компонент для создания нового бота.
  9. Settings.js – это компонент реализующий изменение пароля пользователя.
  10. FunctionBlock.js – это компонент в котором происходит выбор команды для её создания.
  11. Constructor.js – это компонент конструктора в котором реализованы иерархическое отображение структуры бота, добавление команд, открытие окна с изменением команды, откат действий, изменение имени и сохранение бота.
  12. Modal.js – это универсальный компонент для реализации модального окна.
  13. Mail.js – это компонент команды рассылки.
  14. Message.js – это компонент команды сообщения.
  15. ReactList.js – это компонент для реализации блока списка для вызова команды.
  16. FileList.js – это компонент для реализации отображения и взаимодействия с загруженными файлами.

## Руководство пользователя

Главная страница приложения представлена на рисунке 5.1.

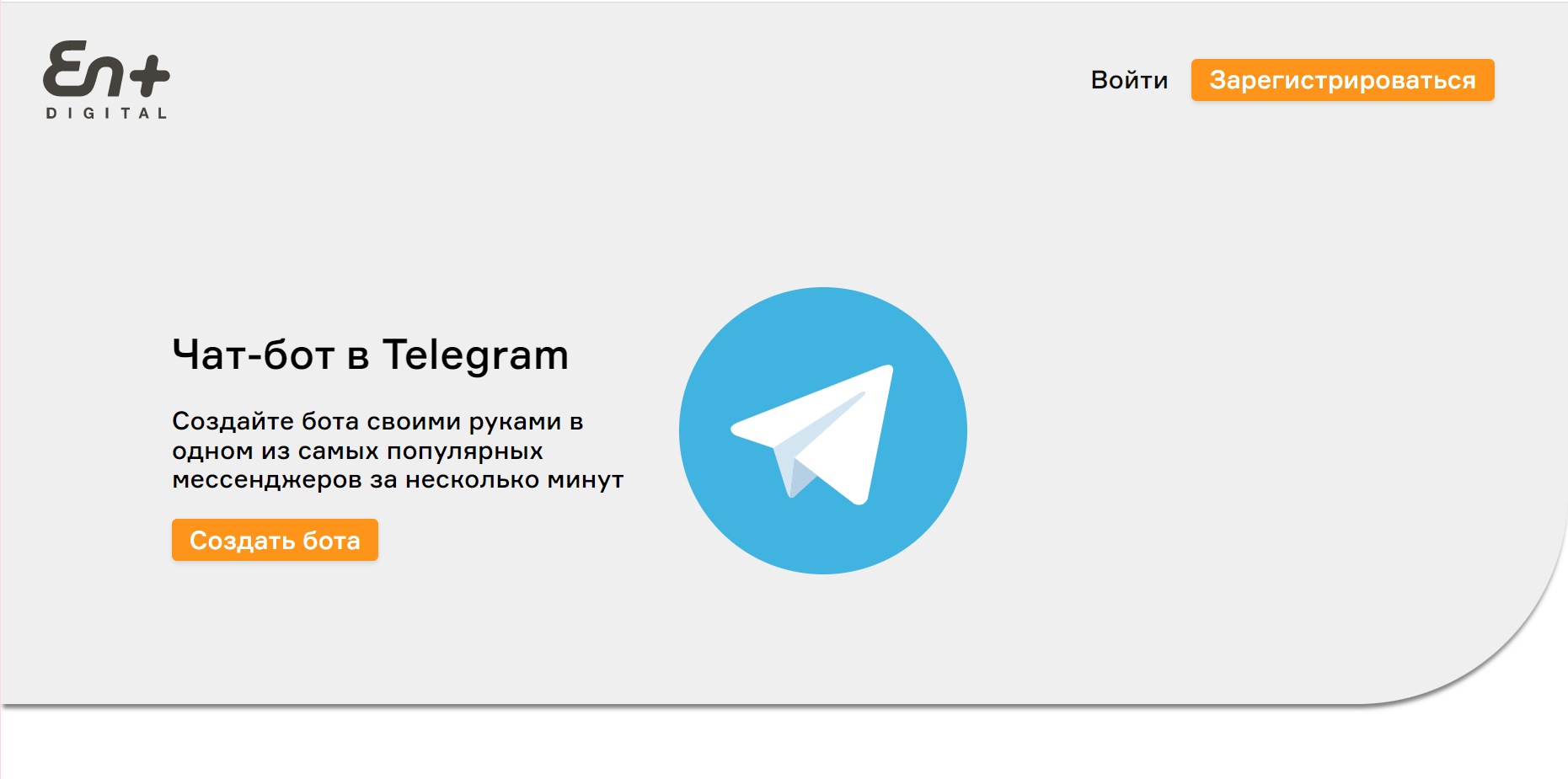


Рисунок 5.1 – Главная страница приложения

### Авторизация

Для использования приложения пользователь должен авторизоваться в системе, для этого необходимо, при отсутствии аккаунта, зарегистрироваться предоставив электронную почту, придумав пароль и имя пользователя (см. рисунок 5.1.1).

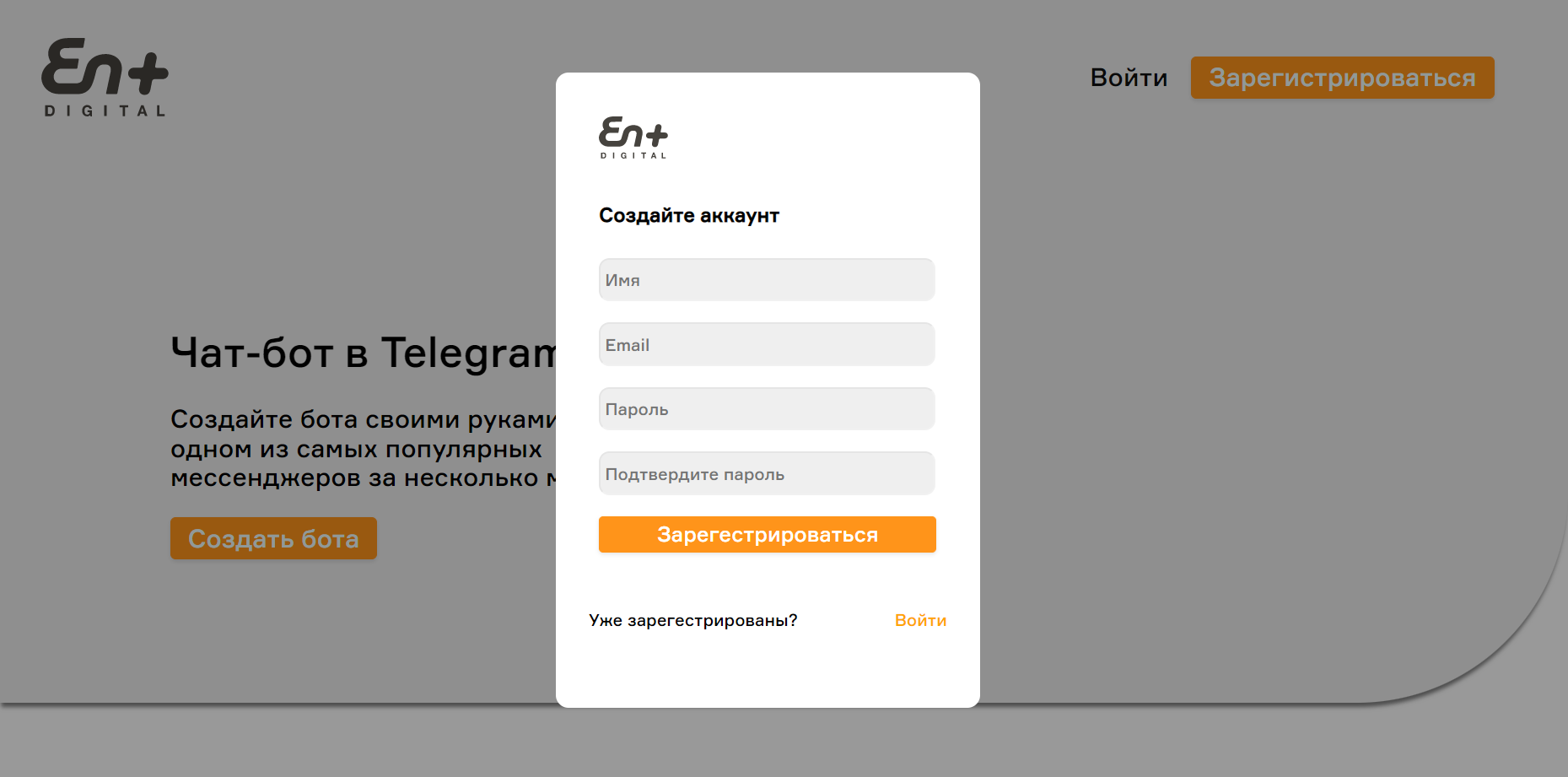


Рисунок 5.1.1 – Форма регистрации

При наличии аккаунта пользователь должен в него войти использую имя пользователя и пароль (см. рисунок 5.1.2).

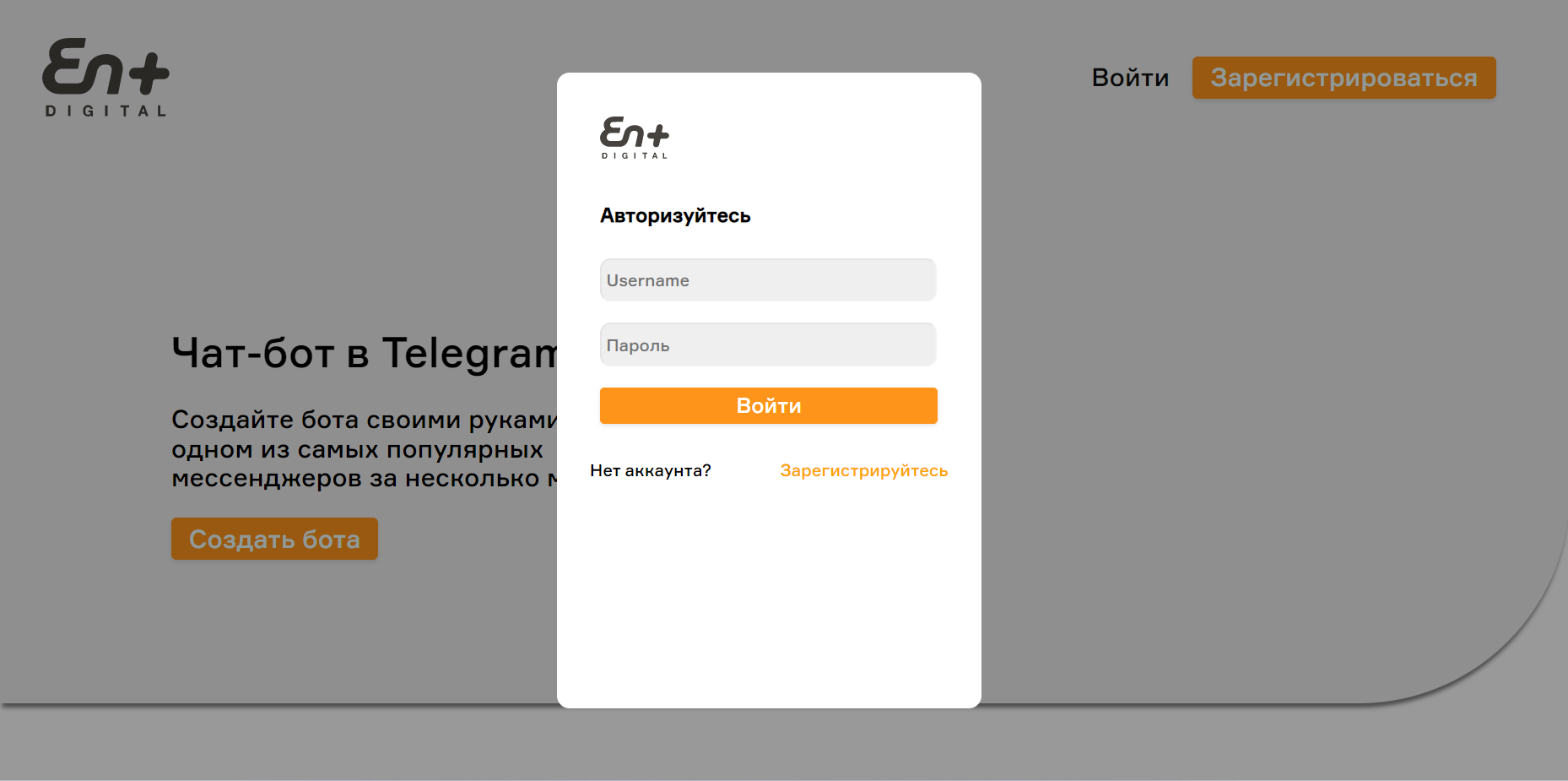


Рисунок 5.1.2 – Форма авторизации

### Список ботов

В окне «Список ботов» (рисунок 5.2.1) пользователь может создать бота, заполнив поле «Имя бота» и нажав на соответствующую кнопку «Создать бота» (при пустом поле имени бота высветится предупреждение).

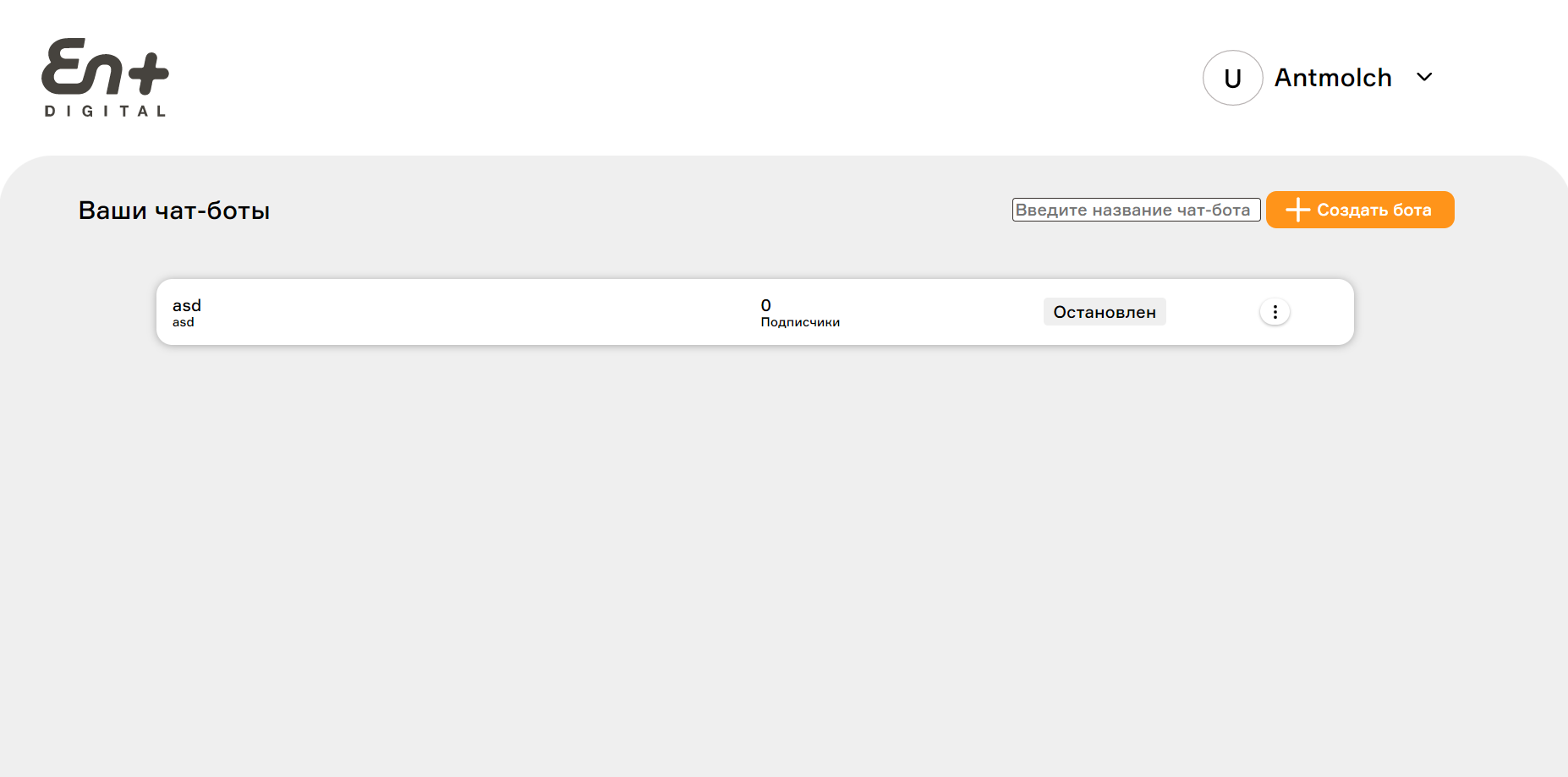


Рисунок 5.2.1 – Список ботов

После чего пользователь попадает в меню руководства по созданию бота (см. рисунок 5.2.2). После выполнения всех пунктов в списке ботов появится созданный бот.

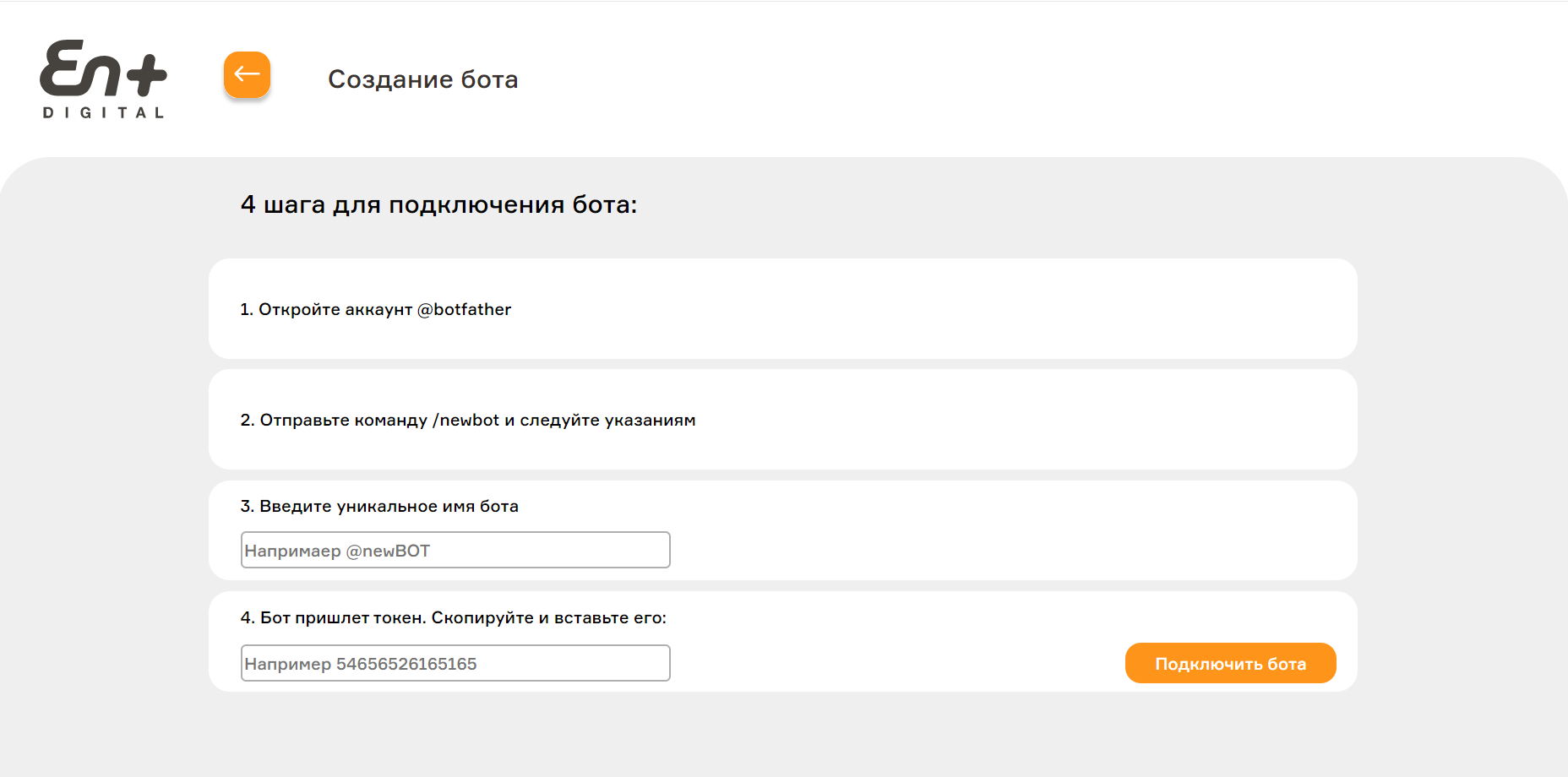


Рисунок 5.2.2 – Меню руководства по созданию бота

Нажав на созданном боте кнопку действий с ботом откроется меню возможных операций (см. рисунок 5.2.3).

**Описание операций:**

* **Изменить** – откроется конструктор бота;
* **Подключить / Отключить** – произведёт включение или отключение бота;
* **Удалить** – безвозвратно удалит бота.

Также по нажатию на поле с название бота откроется его конструктор.

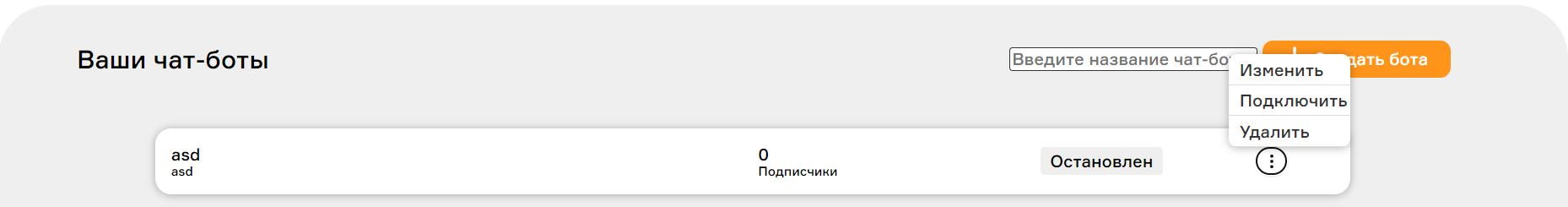


Рисунок 5.2.3 – Меню бота

В заголовке сайта при нажатии на стрелочку справа от имени пользователя откроется меню с возможностью изменения пароля и выходом из аккаунта (см. рисунок 5.2.4).

При выборе пункта изменение пароля на электронную почту пользователя придёт сообщение с ссылкой на страницу изменения пароля.

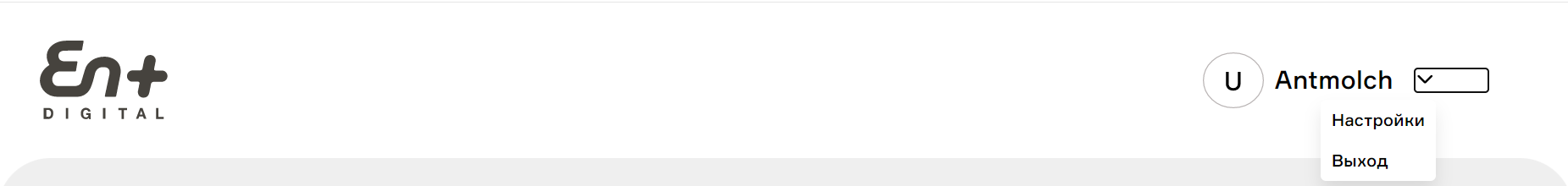


Рисунок 5.2.4 – Меню аккаунта

### Конструктор

Окно конструктора чат-бота представлено на рисунке 5.3.1.

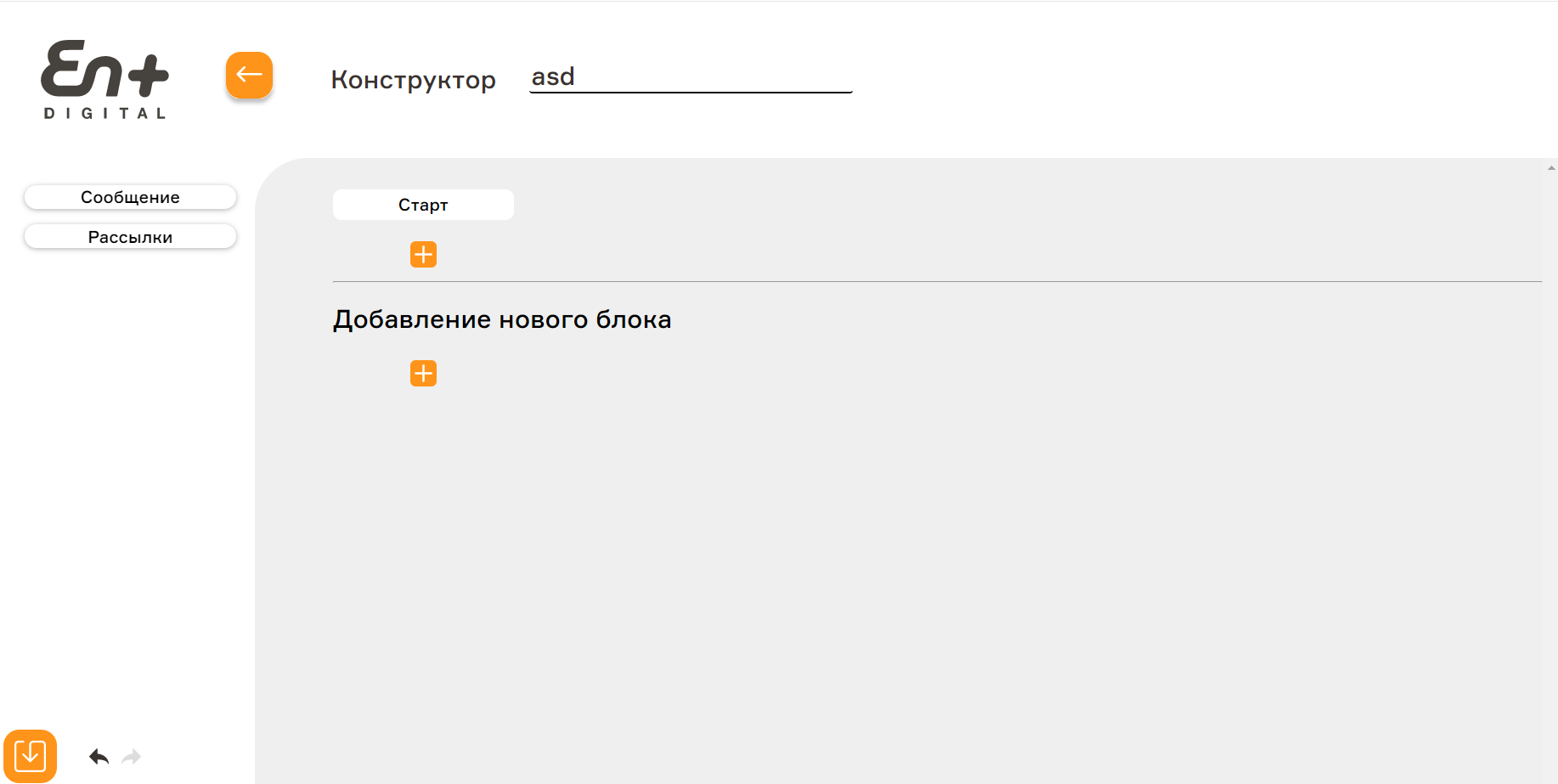


Рисунок 5.3.1 – Окно конструктора

Оно состоит из трёх компонентов: заголовок (шапка сайта), меню команд (левая часть окна) и самого конструктора (центральной серой области).

В компоненте заголовка (см. рисунок 5.3.2). можно вернуться на предыдущую страницу и изменить имя бота



Рисунок 5.3.2 – Компонент заголовка

В компоненте меню команд (см. рисунок 5.3.3). предоставлен выбор из двух типов команд (сообщение и рассылка). Также в нижней части компонента есть кнопки для сохранения бота и отката действий (максимальное количество отмен действий – 10).



Рисунок 5.3.3 – Компонент меню команд

В компоненте конструктора (см. рисунок 5.3.4) представлена структура бота.

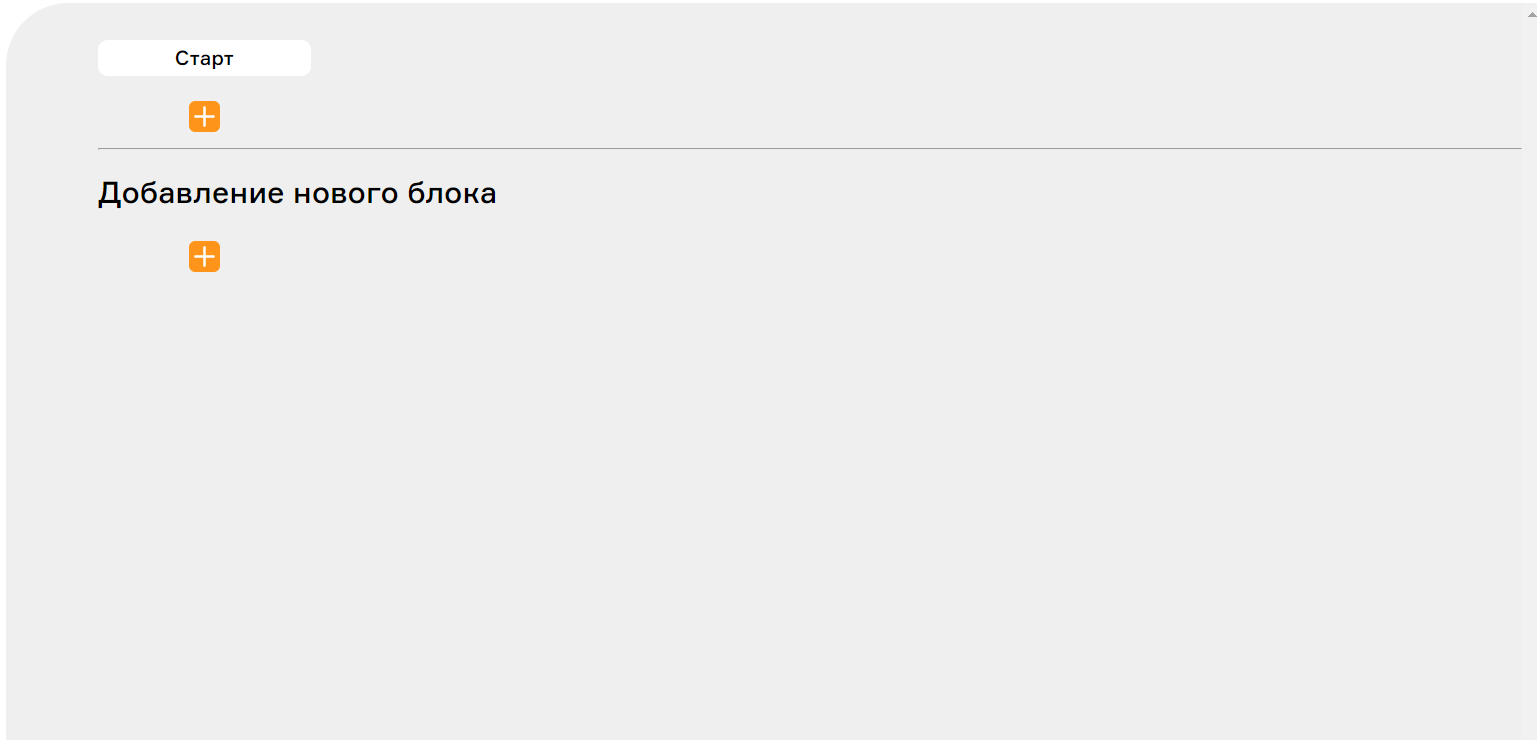


Рисунок 5.3.4 – Компонент конструктора

При выборе типа команды и нажатии на место для добавления (помечено знаком «+»). Произойдёт добавление команды.

**Правила для добавления команд:**

1. Команда рассылки не может быть стартовой командой и не может идти после команды сообщения. Для её добавления необходимо выбрать место под надписью: «Добавление нового блока». После чего она появится в нижней части конструктора с соответствующей подписью (см. рисунок 5.3.5).
2. Команда сообщения может быть добавлена в любое отмеченное место (см. рисунок 5.3.6).

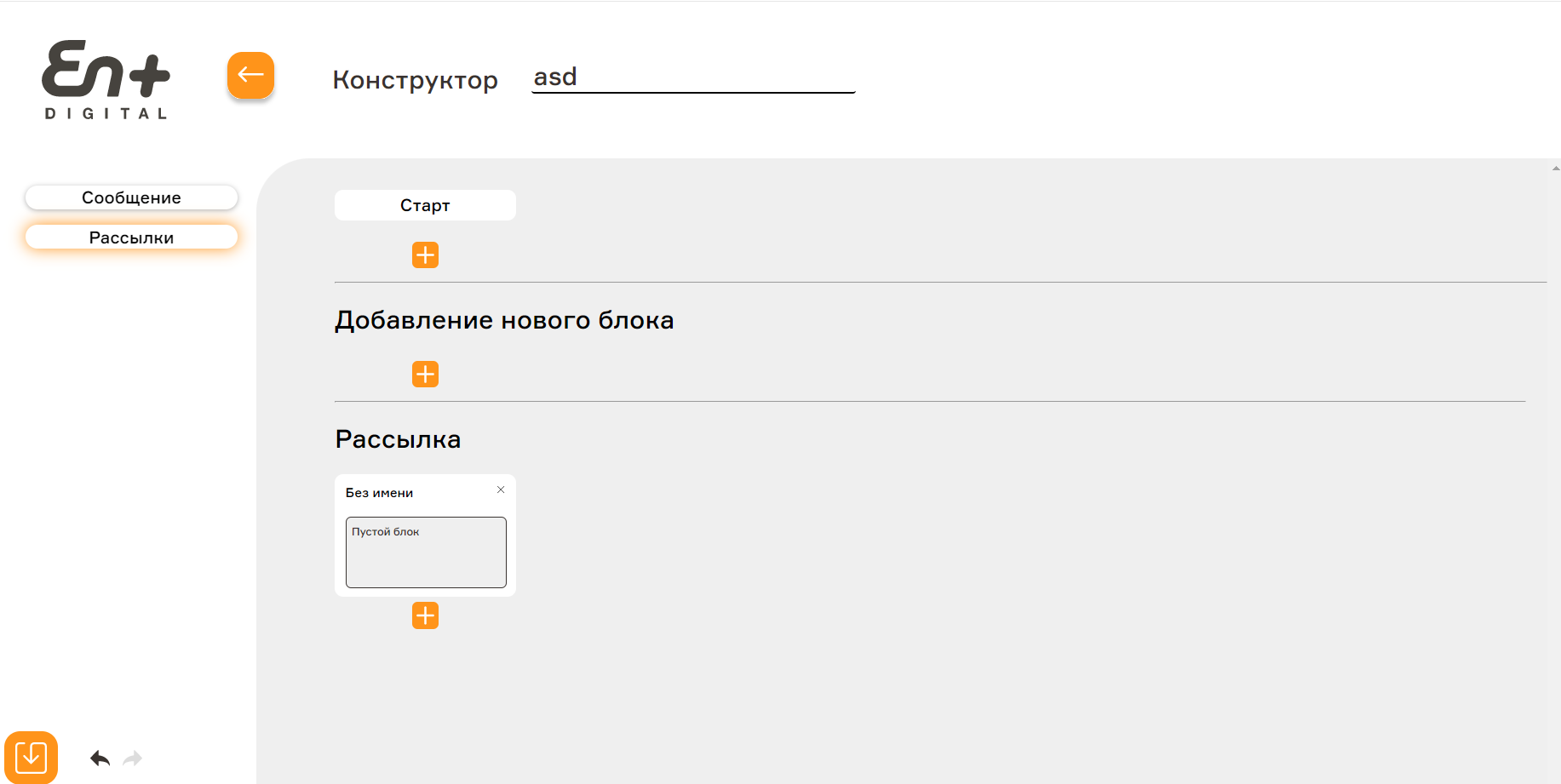


Рисунок 5.3.5 – Добавление команды «Рассылка»

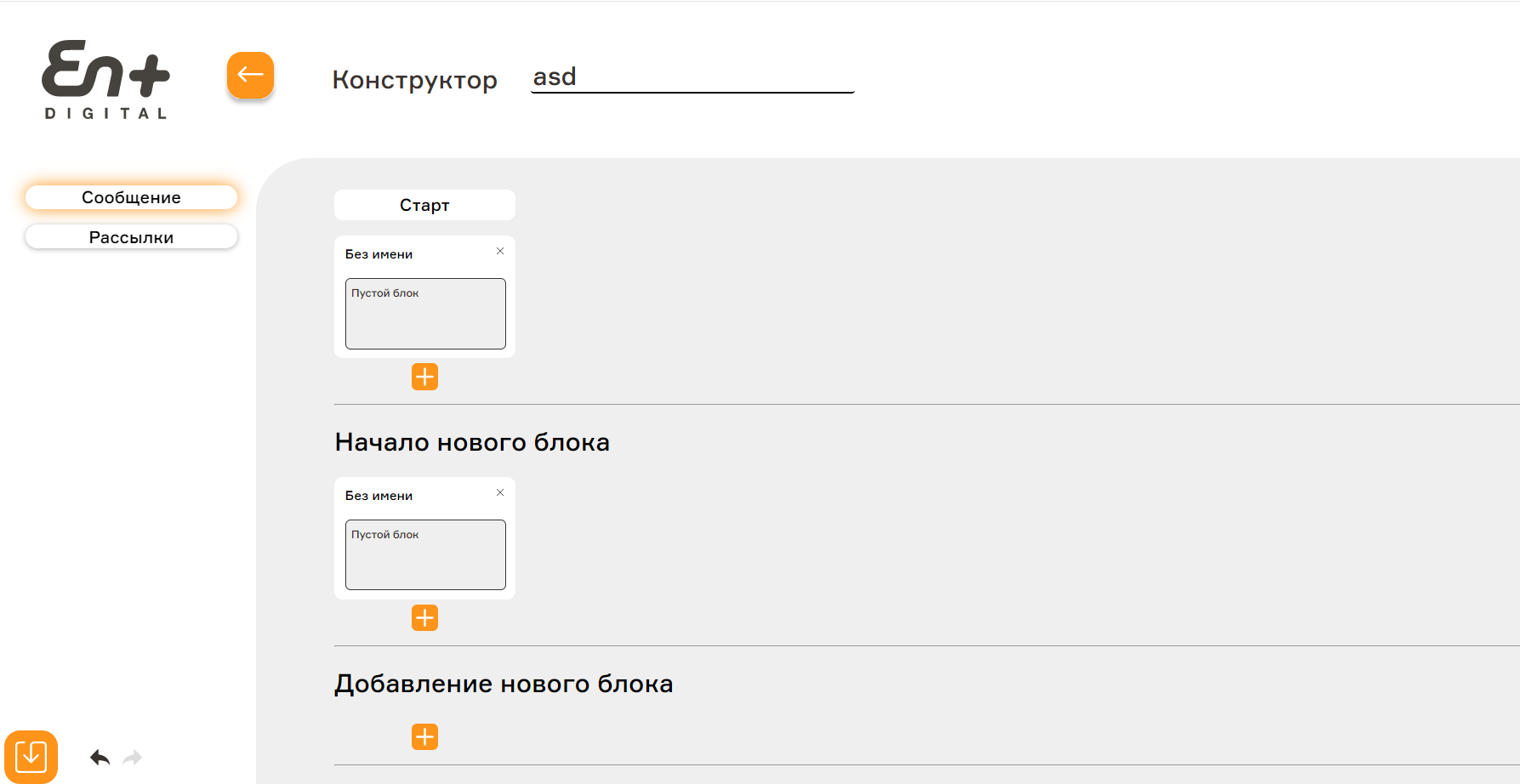


Рисунок 5.3.6 – Добавление команд «Сообщение»

Пример созданной структуры бота представлен на рисунке 5.3.7

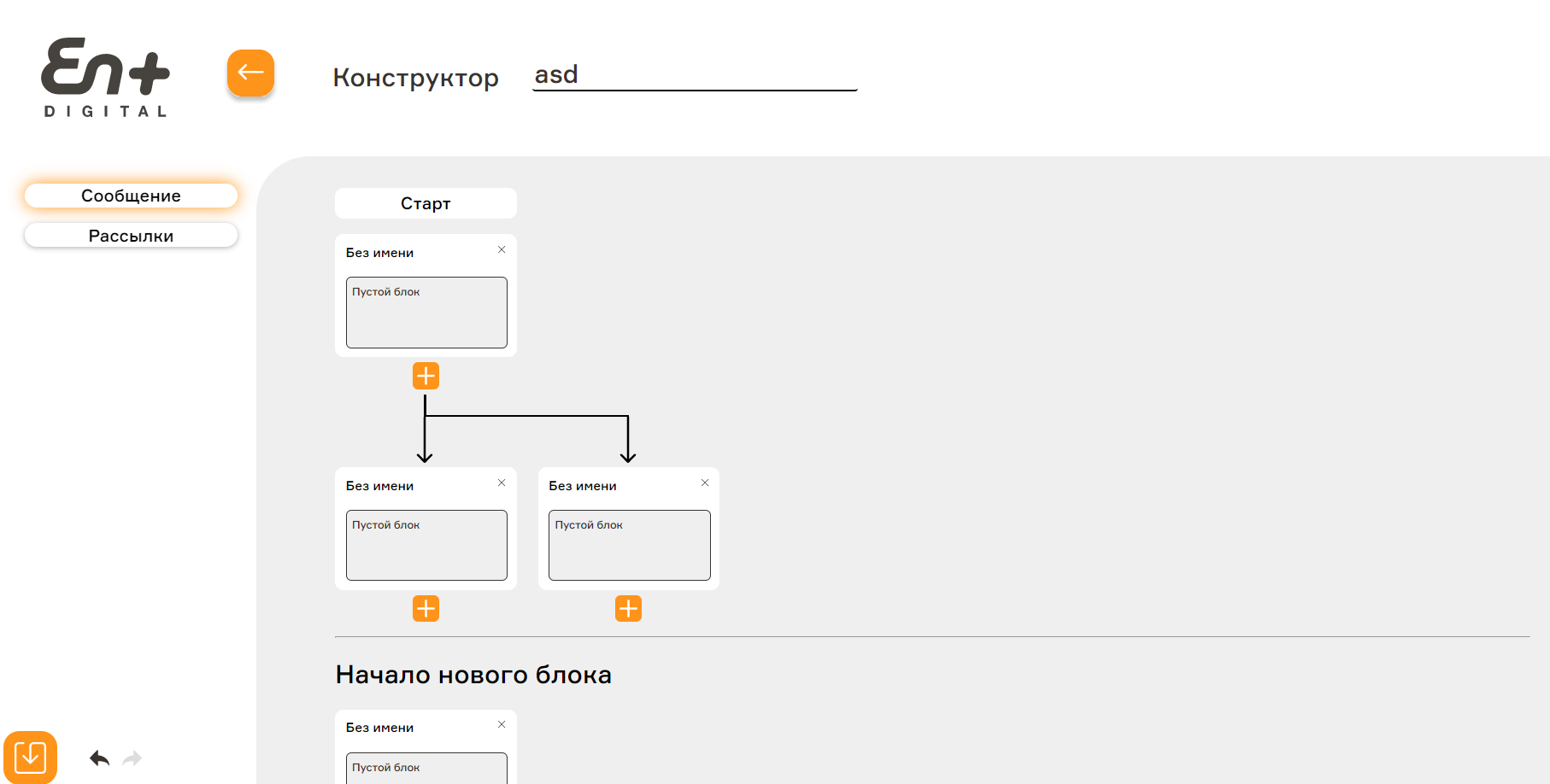


Рисунок 5.3.7 – Пример созданной структуры бота

Для удаления команды нужно нажать на крестик в её правом верхнем углу.

Для изменения команды нужно нажать на неё, после чего всплывёт окно для её изменения. Окно для изменения команды «Сообщение» представлен на рисунке 5.3.8, а для команды «Рассылка» на рисунке 5.3.9.

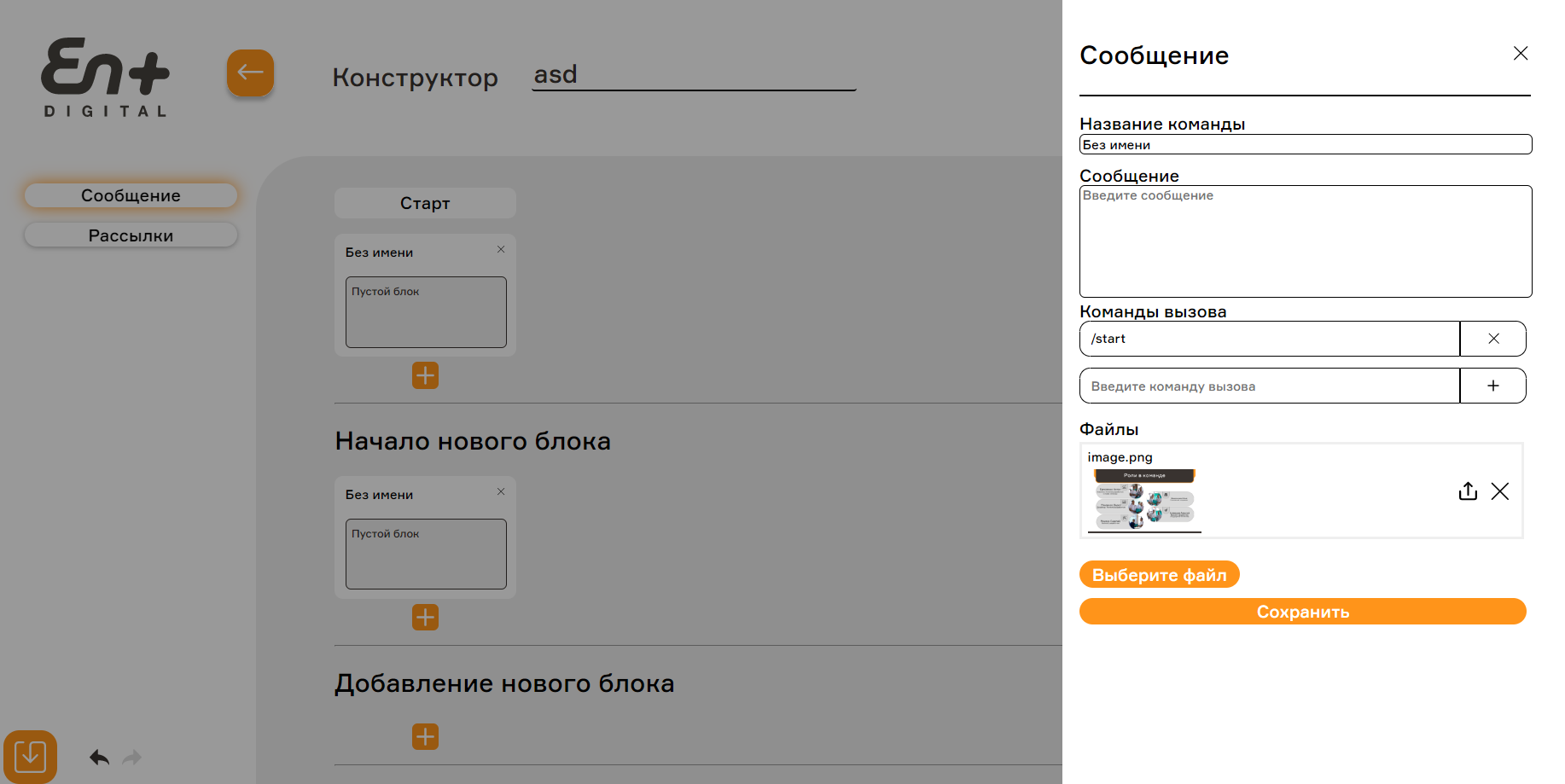


Рисунок 5.3.8 – Окно изменения команды «Сообщение»

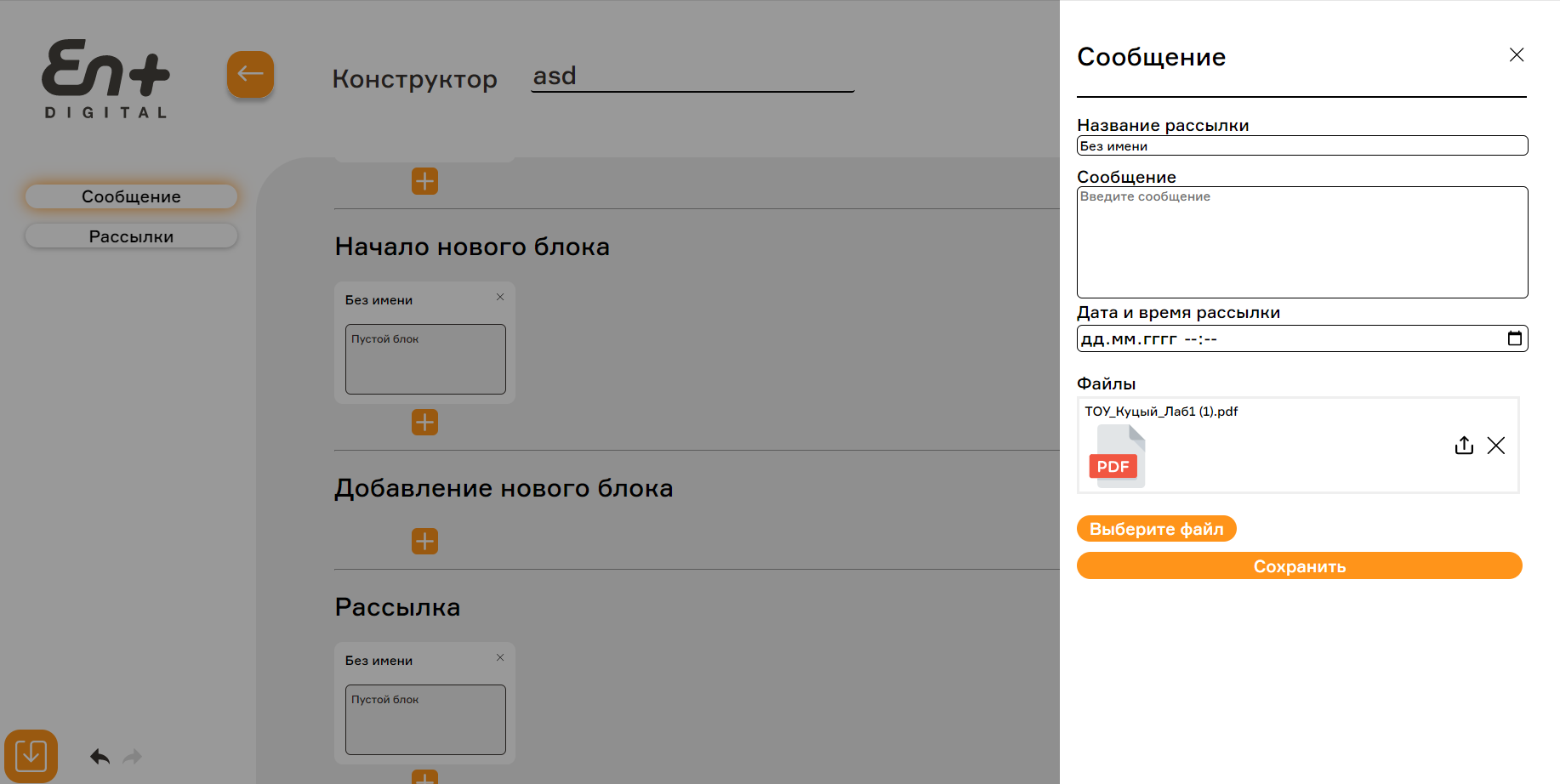


Рисунок 5.3.9 – Окно изменения команды «Рассылка»

Для команды «Сообщение» предлагается изменить её имя, текст сообщения, вызываемые фразы и прикрепить файл формата pdf, docx, mp4, jpeg, png.

Для команды «Рассылка» предлагается изменить её имя, текст сообщения, дату и время вызова и прикрепить файл формата pdf, docx, mp4, jpeg, png.

Ограничения по добавлению файлов – 1 файл не больше 50Мб на команду.

**Для добавления фраз вызова есть ограничения:**

* Вызов не должен совпадать с вызовом другой команды;
* Вызов не должен повторяться.

Для сохранения изменений в команде необходимо нажать на соответствующую кнопку «Сохранить».

Для выхода из окна без сохранения необходимо нажать на крестик ввернем правом углу, либо вне границ окна.

### Проверка работы бота

После создания и подключения бота следует перейти в Telegram и найти своего бота по его уникальному имени (см. рисунок 5.4.1).

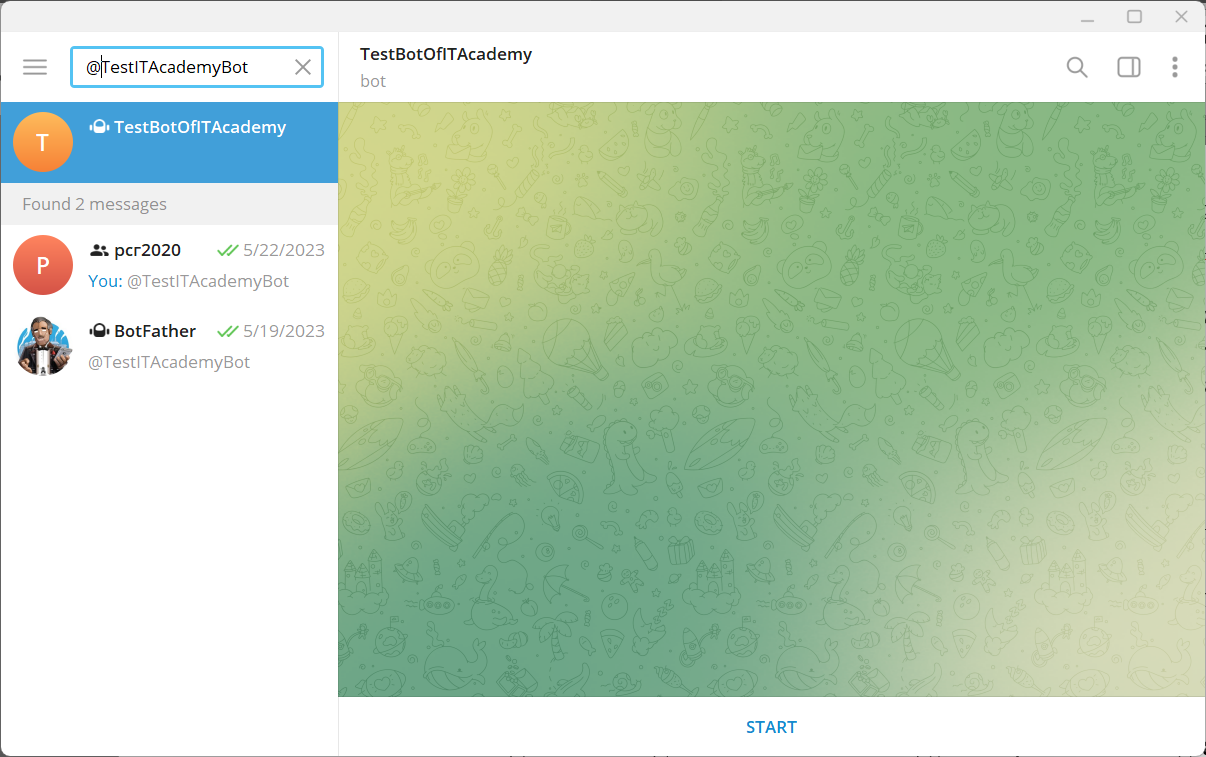


Рисунок 5.4.1 ­ Поиск бота

На рисунке 5.4.2 представлен пример работы Telegram бота, созданного при помощи конструктора.



Рисунок 5.4.2 ­Пример работы бота

## Перспективы

В будущем планируется расширять функциональную часть приложения, а именно:

1. Доработать визуальную часть, чтобы работа с приложением стала полностью интуитивно понятной пользователю.
2. Улучшить масштабируемость.
3. Переработать связь с сервером, для улучшения производительности.
4. Добавить возможность создания ботов для других мессенджеров, например, для Viber.
5. Добавить подробную статистику бота.
6. Добавить новые типы команд.
7. Реализовать нейронную сеть, которая будет определять то, что нужно пользователю по его сообщению.

Веб-приложения "Конструктор чат-ботов" имеет потенциал для монетизации. Можно рассмотреть различные модели монетизации, такие как платные подписки, продажа расширенных функций или услуг, интеграция с рекламными платформами и другие. Успешное коммерческое использование приложения может принести значительную прибыль и стать основой для развития и расширения проекта.

В рамках клиентской части приложения планируется рефакторинг кода для улучшения её скорости работы и облегчения понимания.

# Заключение

Веб-приложение «Конструктор чат-ботов» предлагает гибкий и адаптивный подход к созданию чат-ботов. Это означает, что пользователи могут создавать и настраивать чат-ботов с учетом своих уникальных потребностей и требований. Благодаря гибкости приложения, оно может быть использовано в различных отраслях и сферах деятельности.

В ходе работы были изучены и применены различные технологии и инструменты для создания веб-приложения, способного конструировать и настраивать чат-ботов. Анализ предметной области позволил определить основные требования к приложению и его функциональность.

В ходе разработки приложения были приобретены ценные навыки в области web-программирования, взаимодействия с API, создания пользовательских интерфейсов и разработки серверной части приложения. Работа над проектом позволила практически применить полученные знания и получить опыт разработки полноценного веб-приложения.

# Список использованных источников

1. Документация на React JS – <https://legacy.reactjs.org/docs/getting-started.html> (дата обращения 10.05.2023).
2. Информация о работе с css – <https://developer.mozilla.org/ru/> (дата обращения 12.05.2023).