МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Благовещенский государственный педагогический университет»

Физико-Математический факультет

Кафедра информатики и методики преподавания информатики

Курсовая работа

на тему: Разработки чат-бота для мессенджера «Telegram» информирующей об устройствах, добывающих крипто-валюту

по дисциплине: Технология разработки программного обеспечения

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель:  студент группы 4А | ­­­­­­­­­­­­­­­­  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ дата* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись* | С.А. Либединский |
| Руководитель:  преподаватель | ­­­­­­­­­­­­­­­­  *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ дата* | *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Подпись* | М.А. Серов |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ](#_Toc445457375) 3

[1 Описание средств разработки приложения](#_Toc445457375) 4

[1.1 Язык программирования java 4](#_Toc445457377)

[1.2 Spring 5](#_Toc445457378)

1.3 [JSON 5](#_Toc445457379)

1.4 [Api «Telegram» 6](#_Toc445457379)

[2 Клиент «Telegram» 7](#_Toc445457375)

[2.1 Структура приложения](#_Toc445457377) 7

[2.1.1 UML Диаграмма](#_Toc445457377) 7

[2.2 Описание взаимодействия](#_Toc445457377) 8

[2.3 Создание клиента «Telegram»](#_Toc445457377) 10

[2.3.1 Регистрация клиента в сервисе «Telegram»](#_Toc445457377) 10

[2.3.2 Авторизация в сервисе «Telegram»](#_Toc445457377) 10

[2.3.3 Получение сообщений](#_Toc445457377) 11

[2.3.4 Отправка сообщений](#_Toc445457377) 11

[3 Функции бота 12](#_Toc445457375)

[3.1 Конфигурационный файл](#_Toc445457377) 12

[3.2 Основные команды](#_Toc445457377) 12

[3.2.1 Авторизация](#_Toc445457377) 12

[3.2.1.1 Описание файла конфигурации пользователей](#_Toc445457377) 13

[3.2.2 Прочие команды](#_Toc445457377) 14

[3.2.2.1 Команда START](#_Toc445457377) 15

[3.2.2.2 Команда MONEY](#_Toc445457377) 16

[3.2.2.3 Команда SUBSCRIBE и SUBSCRIBE\_STOP](#_Toc445457377) 17

[ЗАКЛЮЧЕНИЕ](#_Toc445457377) 19

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ](#_Toc445457377) 20

ВВЕДЕНИЕ

Мессенджеры теперь есть в каждом смартфоне, и те, кто не знает, что это - единицы. Мессенджеры стоят на большинстве смартфонах в мире и когда возникает потребность в разработке какого-либо приложения, которые будет выполнять не большое кол-во функций проще всего будет написать бот, на мессенджер, который будет выполнять вашу задумку.

Мессенджеры предоставляют свои api, с помощью которых можно взаимодействовать с ними. Они позволяют легко и быстро разработать бота под их платформу. При этом позволяют использовать любой язык программирования, который позволит использовать их api. Также существуют мессенджеры, которые не предоставляют своё api тем самым загоняя себя в рамки. Эти факторы дают большую помощь при разработки быстрого и легковесного функционала который будет выполнять нужды пользователей.

Целью данной работы и стало создание чат-бота для мессенджера «Telegram».

Объектом работы и являлось чат-бот. А предметом работы послужил язык программирование, api мессенджера «Telegram» и сторонние вспомогательные средства, с помощью которые мы можем создать чат-бота.

Для разработки были поставлены следующие задачи:

* Клиент мессенджера;
* Разработка функционала взаимодействующий с устройствами

1 Описание средств разработки приложения

* 1. Язык программирования java

Для решения выбора языка программирование java повлияло несколько факторов:

* Кроссплатформенность – способность программного обеспечения работать более чем на одной операционной системе;
* Многопоточность – распараллеливания потоков;
* Большое сообщество программистов ‒ большая поддержка языка и огромное количество документации;
* Объектно-ориентированный язык – это позволяет создавать модульные программы, исходный код, который может использоваться многократно.

Java – сильно типизированный объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems (в последующем приобретённой компанией Oracle). Приложения Java обычно транслируются в специальный байт-код, поэтому они могут работать на любой компьютерной архитектуре, с помощью виртуальной Java – машины.

Java представляет собой язык программирования и платформу вычислений, которая была впервые выпущена Sun Microsystems в 1995 г. Существует множество приложений и веб-сайтов, которые не работают при отсутствии установленной Java, и с каждым днем число таких веб-сайтов и приложений увеличивается. Java отличается быстротой, высоким уровнем защиты и надежностью. От портативных компьютеров до центров данных, от игровых консолей до суперкомпьютеров, используемых для научных разработок, от сотовых телефонов до сети Интернет – Java повсюду!

* 1. Spring

Перед выбором технологии на которой будет основываться архитектура, было выбрано Spring – как одна из самых популярных и поддерживаемых.

Spring Framework (или коротко Spring) — универсальный [фреймворк](https://ru.wikipedia.org/wiki/%D0%A4%D1%80%D0%B5%D0%B9%D0%BC%D0%B2%D0%BE%D1%80%D0%BA) [с открытым исходным кодом](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для [Java](https://ru.wikipedia.org/wiki/Java)-платформы. Также существует [форк](https://ru.wikipedia.org/wiki/%D0%A4%D0%BE%D1%80%D0%BA) для платформы [.NET Framework](https://ru.wikipedia.org/wiki/.NET_Framework), названный Spring.NET.

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе [Java](https://ru.wikipedia.org/wiki/Java). Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой [Java Enterprise](https://ru.wikipedia.org/wiki/J2EE), несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

* 1. JSON

Для хранения и передачи информации api «Telegram» использует json что и послужило его использование и распаривание. Достоево JSON:

* Удобочитаемость кода;
* Простота создания объекта данных на стороне сервера;
* Простота обработки данных на стороне клиента;
* Простота расширения;
* Отладка и исправление ошибок;
* Безопасность.

JSON – текстовый формат обмена данными, основанный на JavaScript. Как и многие другие текстовые форматы, JSON легко читается людьми и распознается ПО.

Json Simple - представляет собой простой API для обработки JSON файлов.

Google GSON – представляет собой простой API оборачивающий JSON в классы и обратно.

* 1. Api «Telegram»

API представляет из себя HTTP-интерфейс для работы с ботами в Telegram.

В нем есть следущие возможности:

* Авторизация
* Отправка запросов
* Получение обновлений
* Типы
* Методы

В данной работе использовалась специальная библиотека для java, которая оборачивает api «Telegram» в классы и методы для удобства обращения к серверам.

2 Клиент Telegram

2.1 Структура приложения

Перед разработкой приложение требуется описать структуру приложения.

Рисунок 1 – Структура приложение

Управляющий класс

Клиент

Логика

Сервера Telegram

На Рисунке 1 изображена структура приложения клиента:

* Клиент обрабатывает полученную информацию от сервера или отправляет сообщение серверу;
* Управляющий класс помогает распараллелить работу клиента и логической составляющей приложения;
* Логика осуществляет обработку полученной информации интерпретируя информацию серверу Telegram;
* Сервер Telegram обрабатывает полученную информацию на исключения, и передает ее в чат-бот;

После описание структуры можно приступать к его разработке.

2.1.1 UML Диаграмма

Для удобства создания была создана UML Диаграмма (Рисунок 1).

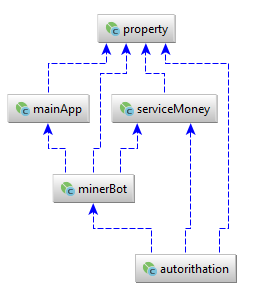


Рисунок 1 – UML диаграмма

* Property – Класс отвечающий за всю информацию находящийся в конфигурационных файлах.
* mainApp – Класс запускающий приложение, а также регистрирующий бота.
* minerBot – Класс который отвечает за авторизацию бота на серверах «Telegram», в том числе отправка и получение сообщений.
* serviceMoney – Класс отвечающий за функционал команд, отправляющий запросы на сервисы, для получения информации о крипто-валюте.
* autorithation – Класс хранящий информацию о пользователе и отвечающий за авторизацию пользователя для доступа к командам.

2.2 Описание взаимодействие

Взаимодействие происходит следующем путём. Пользователь отправляет сообщение боту, сервера Telegram обрабатывают сообщение и отправляют его на подключенный клиент Telegram (Рисунок 2).

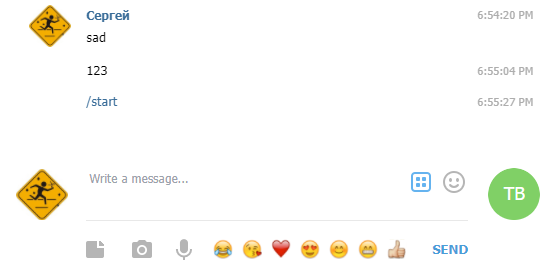


Рисунок 2 – Отправка сообщение боту

После обработки запроса выполняется задуманный разработчиком функционал, в том числе и отправка ответного сообщения (Рисунок 3).

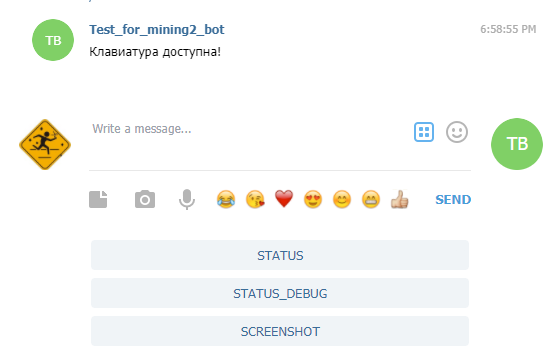


Рисунок 3 – Полученный ответ от бота

2.3 Создание клиента «Telegram»

Для создания клиента «Telegram», потребовалось использовать его api. Api «Telegram» использует стандартные HTTP запросы, поэтому для упрощение работы и отправки запросов на сервера было принято решение использовать разработанную сообществом java библиотеку которая уже описывала взаимодействие с серверами «Telegram». Оставалось лишь изучить методы, описанные в библиотеки и создать свою реализацию клиента.

2.3.1 Регистрация клиента в сервисе «Telegram»

Для того чтобы бот полностью функционировал требуется, зарегистрировать наше подключение. Непосредственно в api «Telegram» есть ряд запросов, но в нашей оболочки под java есть несколько методов которые за это отвечают (Рисунок 4).

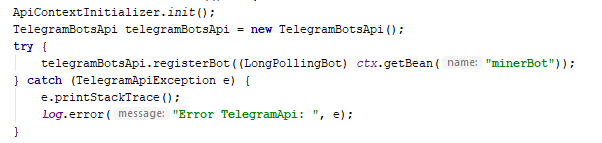


Рисунок 4 - Регистрация бота

Как видно на рисунке в класс «TelegramBotsApi» через метод «registerBot» передается класс, которые описывает необходимые настройки авторизации и методы взаимодействия с серверами «Telegram»

2.3.2 Авторизация в сервисе «Telegram»

Для создание своего бота потребуется выданный ключ и имя бота выданный при его создании. Создание бота происходит непосредственно в самом мессенджере, также с помощью официального бота созданным разработчиками «Telegram» под названием «BotFather».

Непосредственно в коде в переданном классе в «TelegramBotsApi» описываются методы авторизации (Рисунок 5).

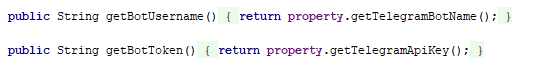


Рисунок 5 – Методы ключа и имени боты

2.3.3 Получение сообщений

Получение сообщений от пользователя происходит в методе «onUpdateRecived», который передает аргумент с полным описанием о том, кто и когда отправил сообщение, а также прочую информацию (Рисунок 6).



Рисунок 6 – Метод получения сообщения

2.3.4 Отправка сообщений

Отправка сообщение происходит с помощью специальных методов с префиксом «Send». Существуют разные методы, например, отправки текста, изображение, видео, музыки и т.п.

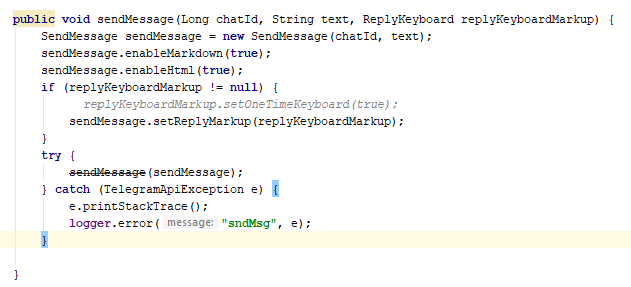


Рисунок 7 – Метод отправки сообщений, с возможностью добавлять специальную клавиатуру.

3 Функции бота

Главной задачей бота уведомлять о состоянии наших устройств, добывающих крипто-валюту. Так как устройства добывали крипто-валюту через определенные сервера, существуют сервисы с открытым api, не сложно было-бы обращаться по идентификационному номеру и запрашивать требуемую информацию.

3.1 Конфигурационный файл

Существует файл с настройками, для удобства быстро сменить какие-либо параметры. В файле указаны имя пользователя, ключ, а также ссылка api сервиса, для отправок запросов (Рисунок 8).



Рисунок 8 – Пример конфигурационного файла

3.2 Основные команды.

3.2.1 Авторизация

Для того чтобы пользователь начал общаться с ботом требовалось ввести пароль (Рисунок 9). Который задавался в специальном конфигурационном файле сервера. Предварительно создав его и описав функционал (Рисунок 10).

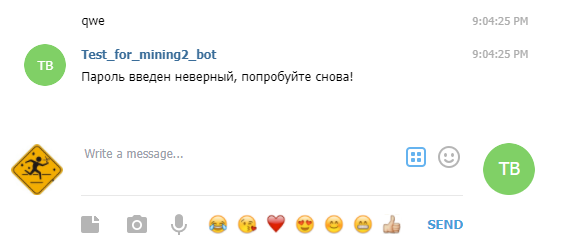
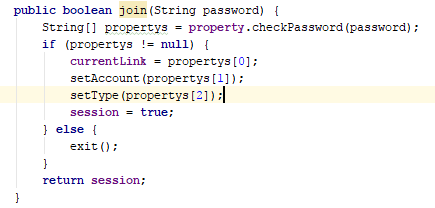


Рисунок 9 – Пример ввода любого сообщения без авторизации.



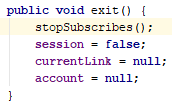


Рисунок 10 – Метод авторизации с аргументом пароля и

де-авторизации

Метод авторизации находится не посредственно в созданном классе «Autorithation», в нем находится уникальный идентификатор пользователя, присланными серверами «Telegram». Тип добываемой крипто-валюты. А также метод де-авторизации (Рисунок 10).

3.2.1.1 Описание файла конфигурации пользователей

Существует файл описания конфигурации пользователей который описывает необходимые параметры для конфигурации функций необходимых для работы бота (Рисунок 11).

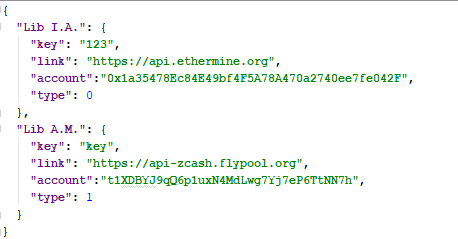


Рисунок 11 – Файл конфигурации

На примере видно первым в корне идут непонятные инициалы - это наименования пользователей. Они условны и нигде не отображаются, необходимы только для удобства конфигурирования.

«key» - Пароль который необходим для авторизации в боте.

«link» - Ссылка на api сайта для запросов.

«account» - Адресс идентификатора крипто-кошелька.

«type» - Тип добываемой валюты, создан для разделения при добычи разной валюты.

3.2.2 Прочие команды

Существуют ряд команд для взаимодействия с ботом.

* START – Предоставляет удобную клавиатуру не требующую ввода команд.
* STATUS – Информирует о статусе устройства.
* STATUS\_DEBUG – Последние 10 строк информации с устройства.
* SCREENSHOT – Изображение с текущего рабочего стола.
* MONEY – Информация с сайта о скорости работы устройств.
* RATE – Текущий курс крипто-валют.
* EXIT – Де-авторизация.
* SUBSCRIBE – Подписка на уведомления от бота по временя команды MONEY.
* SUBSCRIBE\_STOP – Остановить все подписки.
* AVERAGE – Подробная информация о добытой крипто-валюте.
* BALANCE – Текущий баланс крипто-валюты.
* SERVICE\_REBOOT\_RIG – Перезагрузка устройства на которой добывается крипто-валюта.
* SERVICE\_SHUTDOWN\_RIG – Выключение устройства на котором добывается крипто-валюта.

В следующих подпунктах будут подробно расписаны некоторые команды. Не все команды описаны так как не у всех существует реализация, а у многих повторяется и отличается лишь текстом сообщения.

3.2.2.1 Команда START

При отправке данной команды, пользователю предоставляется удобная клавиатура, изображенная на рисунке 12.

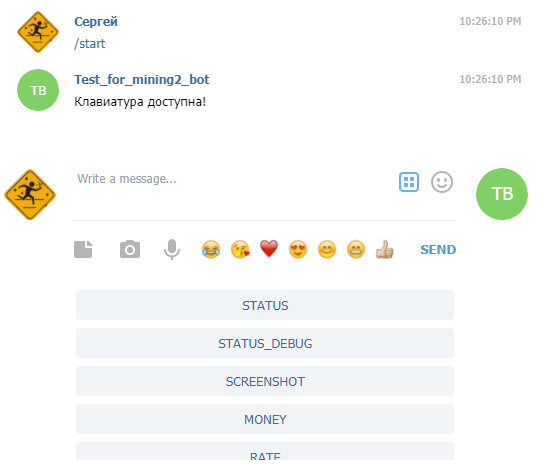


Рисунок 12 – Пример команды START

Для того чтобы клавиатура отображалась, потребовалось при отправке сообщение указать наименование кнопок, при нажатии на которые отправляется сообщение в чат (Рисунок 13).

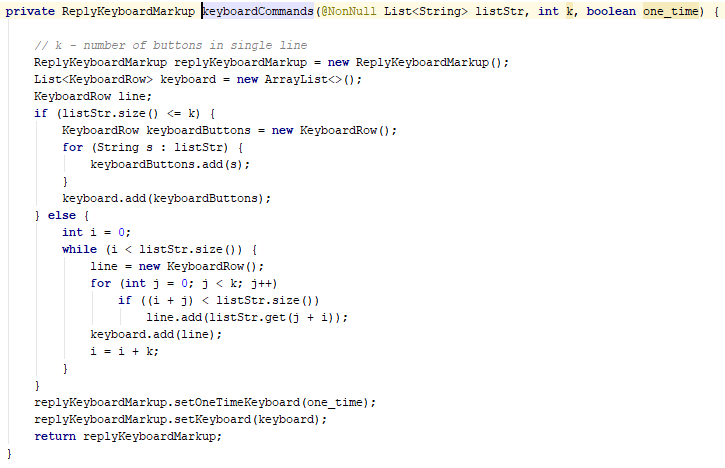


Рисунок 13 – Метод получение клавиатуры с кнопками

3.2.2.2 Команда MONEY

Данная команда работала путем обращения к api сайта, и получении ответа в виде json (Рисунок 14).

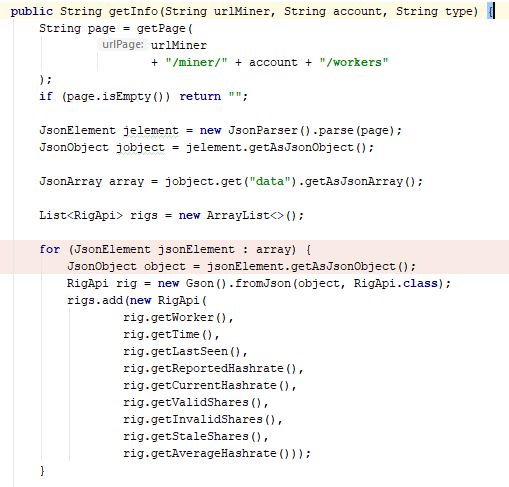


Рисунок 14 – Метод получения информация команды MONEY

В запросе указывался идентификатор, и возвращалась полноценный ответ, после распарсивания которого, сообщение преобразовывалось в ответ в чат-боте (Рисунок 15).



Рисунок 15 – Пример работы команды MONEY

3.2.2.3 Команда SUBSCRIBE и SUBSCRIBE\_STOP

Для реализации данной команды SUBSCRIBE, было задумана клавиатура выбора времени уведомления (Рисунок 16).

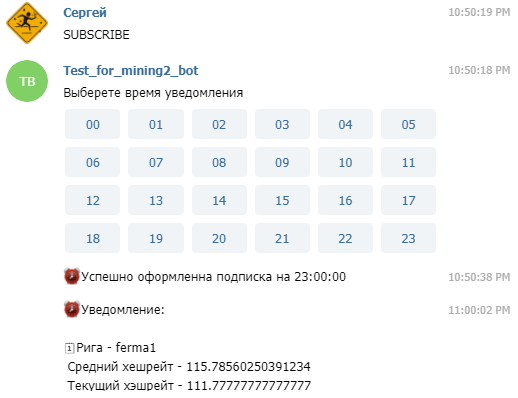


Рисунок 16 – Пример работы команды SUBSCRIBE

Был заведен таймер при достижении времени которого срабатывало уведомление. Реализация которого изображено на рисунке 17.

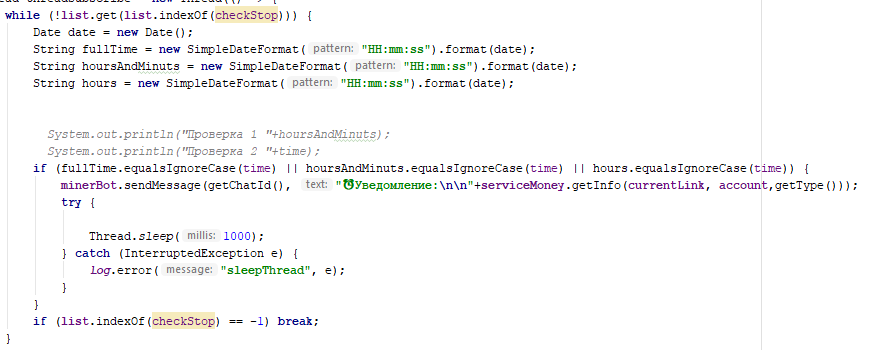


Рисунок 17 – Фрагмент кода таймера для уведомлений

Команда SUBSCRIBE\_STOP останавливает все текущие подписки (Рисунок 18).



Рисунок 18 – Пример команды SUBSCRIBE\_STOP

ЗАКЛЮЧЕНИЕ

В конечном результате мы достигли цели и разработали чат-бот для мессенджера «Telegram» используя современные средства разработки.

Теперь существует возможность быстро и легко создать необходимый функционал, не работая с графической частью нашего продукта, тем самым с экономить время не пренебрегая функционалом.

В дальнейшем можно улучшить свой продукт, используя новые технологии и средства разработки в следствии которых можно сделать более проработанную функционал, добавляя новые возможности и функции. Или использовать чат-бот как маленький блок функций в более масштабном продукте.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Java™ Platform, Standard Edition 8 API Specification: [сайт]. URL: https://docs.oracle.com/javase/8/docs/api/

2. Javadoc for JSON simple: [сайт]. URL: <http://alex-public-doc.s3.amazonaws.com/json_simple-1.1>

3. Telegram Bot API: [сайт]. URL: <https://core.telegram.org/bots/api>

4. Telegram Bot Java Library: [сайт]. URL: <https://github.com/rubenlagus/> TelegramBots

5. Learn Spring: [сайт]. URL: https://spring.io/docs