Компьютерная академия «ШАГ»

Харьковский филиал

Кафедра Разработки программного обеспечения

КУРСОВОЙ ПРОЕКТ

«Telegram – бот “Прогноз погоды”»

Студента группы 30ПР31: Жуков Богдан Валерьевич Научный руководитель:

Колтунов Артем Константинович

Харьков, 2019

**Боты: информация для разработчиков**

Боты — специальные аккаунты в Telegram, созданные для того, чтобы автоматически обрабатывать и отправлять сообщения. Пользователи могут взаимодействовать с ботами при помощи сообщений, отправляемых через обычные или групповые чаты. Логика бота контролируется при помощи HTTPS запросов к нашему [API для ботов](https://tlgrm.ru/docs/bots/api).

Общие сведения

Что могут делать боты?

Вот несколько примеров использования ботов:

* Интеграция с другими сервисами. Например, бот может отправлять комментарии или управлять «умным домом». Или, например, отправлять вам уведомления при совершении каком-то действия или события (Примеры: [GitHub Bot](https://t.me/githubbot), [Image Bot](https://t.me/imagesearchbot)).
* Утилиты и инструменты. Бот может отображать погоду, переводить тексты или предупреждать о предстоящих событиях по вашему запросу (Например: [бот опросов](https://t.me/poll)).
* Одно- и многопользовательские игры. Бот может поиграть с вами в шашки или шахматы, проводить викторины и так далее. (Пример: [Trivia bot](https://t.me/TriviaBot)).
* Социальные сервисы. Бот может находить вам собеседника, основываясь на ваших общих интересах и увлечениях. (Пример: [HotOrBot](https://t.me/hotorbot)).
* Все, что вам захочется. Бота можно запрограммировать для чего угодно. Разве что посуду они помыть не смогут.

Как работают боты?

Как уже было сказано ранее, роботы — особые аккаунты, которые не требуют номера телефона при создании. По сути, эти аккаунты играют роль интерфейса к вашему сервису, который работает на удалённом сервере.

Самое интересное в роботах это то, что для их создания вам не нужно изучать низкоуровневые методы работы с MTProto и шифрованием — общение с роботом организовано при помощи обычного HTTPS интерфейса с упрощёнными методами Telegram API. Мы назвали его [Bot API](https://tlgrm.ru/docs/bots/api).

Как создать бота?

Для этого есть... Бот. Просто напишите пользователю [@BotFather](https://t.me/botfather) и следуйте его инструкциям. Как только вы создали бота и получили свой ключ (токен) авторизации, переходите к настраиванию вашего бота.

Чем бот отличается от обычного аккаунта?

* У роботов нет статусов «онлайн» и «был в сети», вместо этого отображается надпись «бот».
* Для ботов выделено ограниченное место на серверах — все сообщения будут удалены по прошествии определённого срока после обработки.
* Боты не могут сами начать общение с пользователем. Пользователь должен либо добавить робота в группу, либо первым начать с ним диалог. Для этого можно использовать ссылки вида t.me/<bot\_username> или поиск по имени пользователя.
* Имя пользователя у робота должно заканчиваться на «bot» (например, [@controllerbot](https://t.me/controllerbot)).
* При добавлении в конференцию, по умолчанию робот не получает всех сообщений (см. [режим приватности](https://tlgrm.ru/docs/bots" \l "privacy-mode)).
* Роботы никогда не спят, не едят и не жалуются (если только вы не запрограммируете их на обратное).

Суперспособности

У роботов Telegram есть много уникальных возможностей — например, [кастомизированные клавиатуры](https://tlgrm.ru/docs/bots" \l "keyboards), дополнительные интерфейсы для [команд по умолчанию](https://tlgrm.ru/docs/bots" \l "global-commands), [внешнее связывание](https://tlgrm.ru/docs/bots" \l "deep-linking) и специальные [режимы приватности](https://tlgrm.ru/docs/bots" \l "privacy-mode) для групп.

Инлайн-режим

Пользователи могут общаться с вашим ботом напрямую из поля ввода сообщения — из любого чата или группы. Для этого нужно всего лишь набрать имя пользователя вашего бота и запрос для поиска. Получив запрос, бот может возвратить какие-либо результаты. Как только пользователь нажмёт на один из них, он сразу же отправится в чат. Таким образом можно запрашивать контент от бота в чатах, группах или каналах.

Игры

С помощью ботов пользователи могут играть в HTML5-игры в группах или приватных чатах. Игровая платформа Telegram поможет составить таблицу рекордов и оповещать пользователей об изменении рейтинга.

Под капотом у игр — HTML5, поэтому вы можете создавать игры любой сложности. На данный момент командой Telegram созданы несколько демо-игр:

* [Math Battle](https://t.me/gamebot?game=MathBattle)
* [Lumberjack](https://t.me/gamebot?game=Lumberjack)
* [Corsairs](https://t.me/gamebot?game=Corsairs)

Клавиатуры

Одна из самых необычных возможностей Bot API — кастомизированные клавиатуры. При передаче сервером ответа есть возможность передать команду на отображение специальной клавиатуры с предустановленными вариантами ответа (см. [ReplyKeyboardMarkup](https://tlgrm.ru/docs/bots/api" \l "replykeyboardmarkup)). Клиент Telegram, получив сообщение, отобразит пользователю вашу клавиатуру. Нажатие на клавишу сразу же отправит на сервер соответствующую команду. Таким образом можно значительно упростить взаимодействие робота с пользователем. На данный момент для отображения на клавише могут использоваться эмодзи и текст. Вот несколько примеров таких клавиатур:

Команды

Команды представляют собой более гибкий способ общения с ботом. Рекомендуется следующий синтаксис:

/команда [необязательный] [аргумент]

Команда должна начинаться с символа косой черты «/» и не может быть длиннее 32 символов. Команды могут состоять из букв латинского алфавита, цифр и подчёркивания. Несколько примеров:

/get\_messages\_stats

/set\_timer 10min Alarm!

/get\_timezone London, UK

Сообщения, начинающиеся с косой черты, будут всегда доставляться боту (точно также, как и при ответе на его сообщения и на @упоминания бота в чате). Приложения Telegram будут:

* Предлагать список [поддерживаемых команд](https://tlgrm.ru/docs/bots" \l "edit-settings) с их описанием, когда пользователь введёт символ косой черты «/» (чтобы этот пункт работал, вам необходимо задать описание команд у [@BotFather](https://tlgrm.ru/docs/bots" \l "botfather)). Нажатие на описание приведёт к отправке этой команды.
* Показывать кнопку (/) в поле ввода текста во всех чатах с ботами. Нажатие на эту кнопку отобразит список доступных команд.
* Подсвечивать /команды в сообщениях. При нажатии на такую подсвеченную команду, она будет сразу же отправлена боту.

Если в группе есть несколько ботов, вы можете дописать после команды имя бота, чтобы избежать коллизий в общих командах:

/start@TriviaBot

/start@ApocalypseBot

Это происходит автоматически, если вы выбираете команду из списка доступных.

Глобальные команды

Чтобы пользователям было проще работать с ботами, мы просим всех разработчиков реализовывать поддержку нескольких простых команд. В интерфейсе приложений Telegram будут ярлыки (быстрые ссылки) для этих команд.

* /start — начинает общение с пользователем (например, отправляет приветственное сообщение). В эту команду также можно передавать дополнительные аргументы
* /help — отображает сообщение с помощью по командам. Оно может представлять собой короткое сообщение о вашем боте и список доступных команд.
* /settings — (по возможности) возвращает список возможных настроек и команды для их изменения.

При попытке начать общение с роботом, пользователь увидит кнопку СТАРТ. На странице профиля бота также будут доступны ссылки Помощь и Настройки.

Режим приватности

Ботов часто добавляют в группы, чтобы получать различную информацию — новости, уведомления и т.д. Однако, при добавлении бота вы наверняка спрашивали себя: «А что если этот маленький засранец пересылает всю групповую переписку куда-то »налево“?» Именно поэтому у роботов есть режимы приватности.

Робот с включенным режимом приватности не будет получать всех сообщений, а только сообщения, удовлетворяющие этим условиям:

* Сообщения, начинающиеся с символа косой черты "/"
* Сообщения, содержащие *@упоминание* бота
* Ответы на сообщения бота
* Служебные сообщения (о добавлении пользователя, смены изображения группы и т.д.)

Это хорошо со всех сторон: во первых, некоторые люди будут спать спокойно (в своих шапочках из фольги), не опасаясь, что их будут прослушивать. Во-вторых, режим приватности избавляет разработчиков от необходимости обрабатывать сотни ненужных сообщений из групповых чатов.

Режим приватности включен по умолчанию во всех ботах. Он может быть выключен - тогда бот начнёт получать все сообщения, как и обычный пользователь. Всем участникам конференции виден текущий статус режима приватности в списке участников группы.

Мы рекомендуем отключать режим приватности только в случаях крайней необходимости. В подавляющем большинстве случаев, запроса [принудительного ответа](https://tlgrm.ru/docs/bots/api" \l "forcereply) на сообщение бота будет достаточно.

Внешнее связывание

Боты имеют механизм [внешнего связывания](https://ru.wikipedia.org/wiki/Внешнее_связывание), которые позволяет передать дополнительные параметры при запуске. Например, для передачи авторизационного токена пользователя, чтобы соединить его аккаунт с каким-либо внешним сервисом.

У каждого робота есть ссылка, при помощи которой можно начать с ним диалог — https://t.me/<имя бота>. К этой ссылке можно добавить параметры \**start* или startgroup со значением длиной до 64 символов. Пример:

https://t.me/triviabot?startgroup=test

Допустимо использование символов A-Z, a-z, 0-9, \_ и -. Мы рекомендуем использовать [base64url](https://tlgrm.click/wiki-base64) для кодирования параметров.

Переход по ссылке с параметром start начнет чат с ботом, при этом в поле ввода текста будет отображаться кнопка СТАРТ. При переходе по ссылке с параметром startgroup пользователю будет предложено добавить бота в одну из доступных групп. Как только пользователь подтвердит своё действие (нажмёт на кнопку СТАРТ или выберет группу), бот получит сообщение следующего формата:

/start PAYLOAD

В PAYLOAD будет значение параметра start или startgroup, который был передан в ссылке.

Пример реализации внешнего связывания

Предположим, сайт example.com хочет отправлять пользователю уведомления с помощью Telegram-бота. Ниже приведено описание последовательности действий для реализации отправки уведомлений пользователю с ID 123.

1. [Создайте](https://tlgrm.ru/docs/bots" \l "botfather) бота с любым именем, например, @ExampleComBot
2. Настройте [вебхук](https://tlgrm.ru/docs/bots/api" \l "setwebhook) для входящих сообщений
3. Сгенерируйте ключ достаточной длины, напр. $memcache\_key = 'TESTKEY'
4. Запишите значение 123 с ключом $memcache\_key в Memcache на 3600 секунд (один час)
5. Отправьте пользователю ссылку https://t.me/ExampleComBot?start=TESTKEY
6. Настройте обработчик веб-хуков так, чтобы он запрашивал в Memcached значение входящего параметра каждый раз, когда приходит сообщение, начинающееся со /start. Если ключ существует, записываем chat\_id, переданный в веб-хук, в telegram\_chat\_id для пользователя 123. Удаляем ключ из Memcache.
7. Теперь, если мы хотим отправить сообщение пользователю 123, сначала проверяем telegram\_chat\_id. Если оно существует, используем метод [sendMessage](https://tlgrm.ru/docs/bots/api" \l "sendmessage), чтобы отправить пользователю сообщение.

BotFather

[BotFather](https://t.me/botfather) — один бот, чтобы править всеми. При помощи него меняются настройки у существующих ботов и создаются новые.

Создание бота

Напишите команду /newbot, чтобы создать нового робота. BotFather спросит у вас имя нового бота и предложит придумать username.

Имя (name) будет отображаться в контактах и чатах.

Username — короткое имя на латинице, которое используется для упоминаний бота и в ссылках на профиль в telegram.me. Username должен состоять из букв латинского алфавита, подчёркиваний и цифр и быть длиной от 5 до 32 символов. Также имя пользователя обязательно должно заканчиваться на «bot», например: «tetris\_bot» или «TetrisBot».

Ключ (токен) это набор символов вида 110201543:AAHdqTcvCH1vGWJxfSeofSAs0K5PALDsaw, который нужен, чтобы получать и отправлять сообщения с помощью [Bot API](https://tlgrm.ru/docs/bots/api).

Создание токена

Если вы потеряли или утратили доступ к токену, отправьте команду /token, чтобы сгенерировать новый.

Настройки

* /setname — Изменить имя робота.
* /setdescription — Изменить описание робота, представляющее собой короткий текст с описанием бота. Пользователи увидят его в самом начале, под заголовком «Что умеет этот робот?».
* /setabouttext — Изменить информацию о боте, ещё более короткий текст, отображающийся в профиле бота. Ещё, если кто-то поделится вашим ботом, то вместе со ссылкой на него отправится этот текст.
* /setuserpic — Изменить аватарку бота. Картинки — всегда хорошо.
* /setcommands — Изменить список команд бота. Каждая команда состоит из собственно командного слова, начинающегося с символа косой черты («/») и короткого описания. Пользователи увидят список команд при вводе символа «/».
* /setjoingroups — Определяет, можно ли добавлять вашего бота в группы.
* /setprivacy — Определяет, все ли сообщения видит ваш бот в группах. В выключенном состоянии роботу будут отправляться все сообщения.
* /deletebot — Удалить бота и его имя пользователя.

В Интернете много сайтов, где можно создать своего бота с уже готовым деплоем. Но дело в том. что в своем большинстве создаются боты, которые могут предоставить справочную информацию и прочее. Мой бот - приложение, к которому можно привязать базу данных, выполнять запросы на различные API, парсить сайты. Он будет выдавать прогноз погды по запросу. Я надеюсь что в этих строках я немного разъяснил вам о чем собираюсь писать.

Бота в Telegram зарегистрировать очень просто, этот процесс подробно описан в документации по ссылке выше. Для нашего приложение необходимо знать только имя бота и токен, который вы получите при регистрации.

По сути бот — просто консольное веб-приложение. Никакого фронтенда, чистая обработка команд. Если вы желаете хорошо освоить Hibernate или научиться парсить JSON, то такой проект для вас.

Начнем с того чтобы подключить зависимость в pom.xml (подразумеваем что вы используете Maven). Сделать это можно так:

<dependency>

<groupId>org.telegram</groupId>

<artifactId>telegrambots</artifactId>

<version>3.6</version>

</dependency>

Затем создаем класс Bot, унаследуем его от класса TelegramLongPollingBot, переопределив его методы:

**public** **class** Bot **extends** TelegramLongPollingBot {

/\*\*

\* Метод для приема сообщений.

\* @param update Содержит сообщение от пользователя.

\*/

@Override

**public void onUpdateReceived(Update update) {**

**Model model = new Model();**

**Message message = update.getMessage();**

**if (message != null && message.hasText()) {**

**switch (message.getText()) {**

**case "/help":**

**sendMsg(message, "Чем могу помочь?");**

**break;**

**case "/setting":**

**sendMsg(message, "Что будем настраивать?");**

**break;**

**default:**

**try {**

**sendMsg(message, Weather.getWeather(message.getText(), model));**

**} catch (IOException e) {**

**sendMsg(message, "Город не найден!");**

**}**

**}**

**}**

**}**

/\*\*

\* Метод для настройки сообщения и его отправки.

\* @param chatId id чата

\* @param s Строка, которую необходимот отправить в качестве сообщения.

\*/

**public void sendMsg(Message message, String text) {**

**SendMessage sendMessage = new SendMessage();**

**sendMessage.enableMarkdown(true);**

**sendMessage.setChatId(message.getChatId().toString());**

**sendMessage.setReplyToMessageId(message.getMessageId());**

**sendMessage.setText(text);**

**try {**

**setButtons(sendMessage);**

**sendMessage(sendMessage);**

**} catch (TelegramApiException e) {**

**e.printStackTrace();**

**}**

**}**

/\*\*

\* Метод возвращает имя бота, указанное при регистрации.

\* @return имя бота

\*/

@Override

**public** String getBotUsername() {

**return** "WeatherBot";

}

/\*\*

\* Создаем и настраиваем кнопки

\* @return token для бота

\*/

@Override

**public** String getBotToken() {

**return** "829339813:AAEQGPAo872xkiNnXs6-uZlIdpp4AUFITNk";

}

}

/\*\*

\* Создаем и настраиваем кнопки

\*/

public void setButtons(SendMessage sendMessage) {

ReplyKeyboardMarkup replyKeyboardMarkup = new ReplyKeyboardMarkup();

sendMessage.setReplyMarkup(replyKeyboardMarkup);

replyKeyboardMarkup.setSelective(true);

replyKeyboardMarkup.setResizeKeyboard(true);

replyKeyboardMarkup.setOneTimeKeyboard(false);

List<KeyboardRow> keyboardRowList = new ArrayList<>();

KeyboardRow keyboardFirstRow = new KeyboardRow();

keyboardFirstRow.add(new KeyboardButton("/help"));

keyboardFirstRow.add(new KeyboardButton("/setting"));

keyboardRowList.add(keyboardFirstRow);

replyKeyboardMarkup.setKeyboard(keyboardRowList);

}

Ну и содержимое метода main:

**public static void main(String[] args) {**

**ApiContextInitializer.init();**

**TelegramBotsApi telegramBotsApi = new TelegramBotsApi();**

**try {**

**telegramBotsApi.registerBot(new Bot());**

**} catch (TelegramApiRequestException e) {**

**e.printStackTrace();**

**}**

**}**

Вписав в методы *getBotUsername()* и *getBotToken()* мы запускаем бота. Пока он только перенаправляет нам любые сообщения, которые мы отправим ему, этакое «зеркало». Работает это все следующим образом: когда вы запускаете приложение, оно начинает раз в n количество секунд отправлять на сервер Telegram GET запрос по следующему URL: https://api.telegram.org/BotToken/getMe, где BotToken – токен вашего бота, получая в ответ JSON, в котором находятся все сообщения. Каждое такое сообщение обрабатывается библиотекой и приходит в метод *OnUpdateReceived(Update update)* объектом *Update*. С ним то мы и работаем. В этом вся прелесть Telegram-ботов, они могут работать на любом компьютере, для тестирования нужно просто запустить приложение, не нужно деплоить его на хостинг после каждого изменения. Это очень удобно.

То как обрабатывать сообщения и что отправлять в ответ ограничено только лишь средствами языка и библиотекой, все остальное на ваше усмотрение. Вы можете сделать бота, который будет искать для вас видео на YouTube, можете сделать бота, который каждый день будет присылать вам то, что вы отправите себе, к примеру, за год, эдакую капсулу времени. А можете научиться интегрироваться к CRM-системам и делать ботов для малого бизнеса, все ограничено вашей фантазией.

Идем дальше. Те, кто пользовался ботами знают, что с ними удобно взаимодействовать командами, начинающимися со знака «/», например /start. Но есть способ удобнее — кнопки. Есть два вида кнопок: те, что появляются под полем ввода, ReplyKeyboardMarkup и кнопки, которые находятся непосредственно под сообщением, к которому привязаны, InlineKeyboardMarkup. В документации вы можете поверхностно ознакомиться с их описанием. **ReplyKeyboardMarkup.**

По сути это — массив массивов кнопок, List <KeyboardRow <KeyboardButton>>. Вот пример кода, который создает клавиатуру

**public** **synchronized** **void** setButtons(SendMessage sendMessage) {

// Создаем клавиуатуру

ReplyKeyboardMarkup replyKeyboardMarkup = **new** ReplyKeyboardMarkup();

sendMessage.setReplyMarkup(replyKeyboardMarkup);

replyKeyboardMarkup.setSelective(true);

replyKeyboardMarkup.setResizeKeyboard(true);

replyKeyboardMarkup.setOneTimeKeyboard(false);

// Создаем список строк клавиатуры

List<KeyboardRow> keyboard = **new** ArrayList<>();

// Первая строчка клавиатуры

KeyboardRow keyboardFirstRow = **new** KeyboardRow();

// Добавляем кнопки в первую строчку клавиатуры

keyboardFirstRow.add(**new** KeyboardButton(“Привет”));

// Вторая строчка клавиатуры

KeyboardRow keyboardSecondRow = **new** KeyboardRow();

// Добавляем кнопки во вторую строчку клавиатуры

keyboardSecondRow.add(**new** KeyboardButton(“Помощь”);

// Добавляем все строчки клавиатуры в список

keyboard.add(keyboardFirstRow);

keyboard.add(keyboardSecondRow);

// и устанваливаем этот список нашей клавиатуре

replyKeyboardMarkup.setKeyboard(keyboard);

}

В методе *sendMsg()* мы вызываем этот метод, передавая ему сообщение, таким образом устанавливая для такого сообщения клавиатуру. Когда мы отправим это сообщение пользователю, то он увидит текст сообщения, который мы установили, а также 2 кнопки, на которых будет написано Привет и Помощь, друг под дружкой.

По нажатию на эти кнопки боту будет отправлено сообщение, текст которого представляет собой то, что написано на кнопке. То есть если клиент нажмет «Помощь», то боту придет сообщение с текстом “Помощь“. Для него это как будто бы клиент сам написал текст “Помощь“ и отправил бы ему. Ну а затем вы обрабатываете такие сообщения.

**InlineKeyboardMarkup.**

Это тоже массив массивов, он похож на предыдущий Markup, но логика работы здесь немного другая. Такая клавиатура привязывается к определенному сообщению и существует только для него. Вот метод для установки Inline-клавиатуры

**private** **void** setInline() {

List<List<InlineKeyboardButton>> buttons = **new** ArrayList<>();

List<InlineKeyboardButton> buttons1 = **new** ArrayList<>();

buttons1.add(**new** InlineKeyboardButton().setText(“Кнопка“).setCallbackData(17));

buttons.add(buttons1);

InlineKeyboardMarkup markupKeyboard = **new** InlineKeyboardMarkup();

markupKeyboard.setKeyboard(buttons);

}

Создаем *List* в *List*, добавляем в первую строку Inline-кнопку. Такая кнопка может содержать URL, ссылку на канал или же *CallbackQuery*, о которой я напишу чуть позже. Здесь мы устанавливаем текст для нашей кнопки, который будет видеть пользователь, а затем устанавливаем данные, которые будут отправлены боту. В нашем примере пользователь видит «Привет», а боту при нажатии отправится число 17, это и есть наш *CallbackQuery*.

Пару слов о C*allbackQuery*. Для получения таких данных из объекта *Update* нужно *выполнить update.getCallbackQuery()*, этот метод возвращает *CallbackQuery*, из которого уже можно получить данные, переданные боту. Не нужно пытаться получить эти данные через метод *update.getMessage().getText()*, получите *NullPointerException*.

@Override

**public** **void** onUpdateReceived(Update update) {

**if**(update.hasMessage()) {

ThreadClass thread = **new** ThreadClass(update.getMessage());

} **else** **if**(update.hasCallbackQuery()) {

AnswerCallbackThread answerThread = **new** AnswerCallbackThread(update.getCallbackQuery());

}

}

Если есть сообщение, отправляем на обработку в новый поток сообщение, если есть *CallbackQuery*, отправляем его на обработку в соответствующий поток.

На *CallbackQuery* можно отправлять ответ. У каждого объекта в Telegram есть свой id. Для отправки ответа на определенный *CallbackQuery* нужно знать лишь его id, который мы получим из соответствующего объекта. Для отправки ответа вызовем такой метод:

**public** **void** answerCallbackQuery(String callbackId, String message) {

AnswerCallbackQuery answer = **new** AnswerCallbackQuery();

answer.setCallbackQueryId(callbackId);

answer.setText(message);

answer.setShowAlert(true);

**try** {

answerCallbackQuery(answer);

} **catch** (TelegramApiException e) {

e.printStackTrace();

}

}

ВАЖНО: Текст в ответе на *CallbackQuery* должен быть не длиннее 200 символов!

При отправке такого ответа клиент получит всплывающее окно, в котором будет написано сообщение. Такое окно может исчезнуть через несколько секунд само после появления, а может висеть до тех пор, пока пользователь не нажмет ок. Для переключения этих режимов мы вызываем метод *answer.setShowAlert(true).* При *true* окошко висит до нажатия ок, при *false* исчезает через 5 секунд.

В итоге у меня получился бот который выдает прогноз погоды любого города по запросу пользователься. Бот выводит: температуру, влажность, какая погода. Функционал у бота не очень большой, но он работает правильно и стабильно.