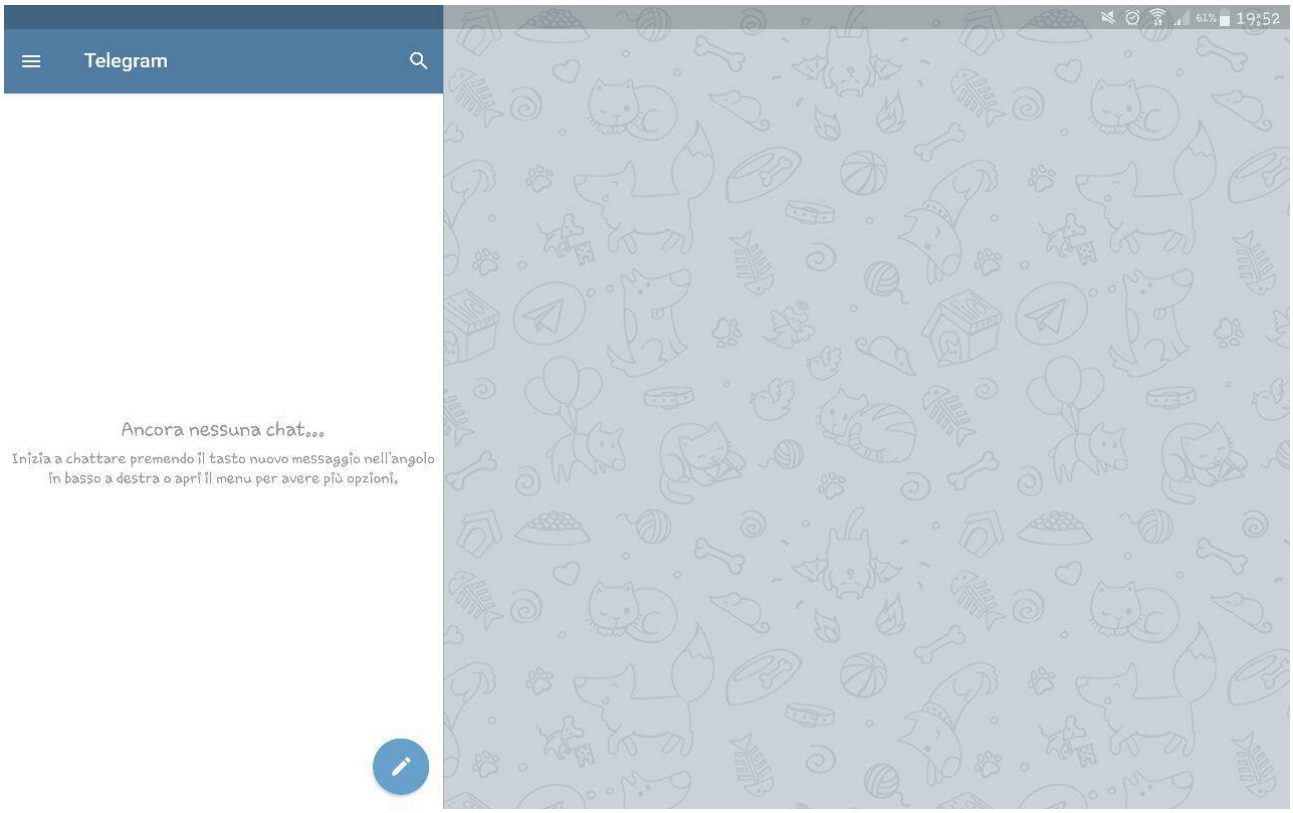
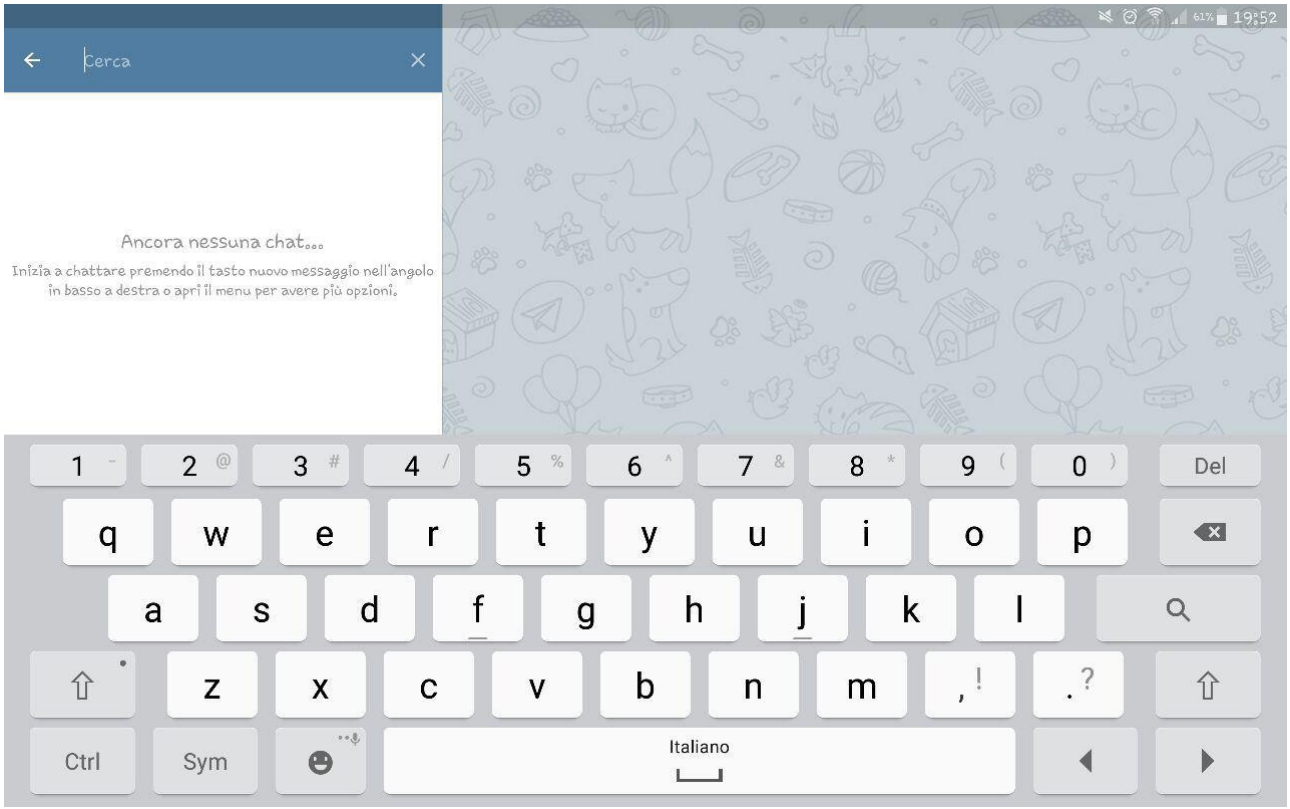
FISHGRAM

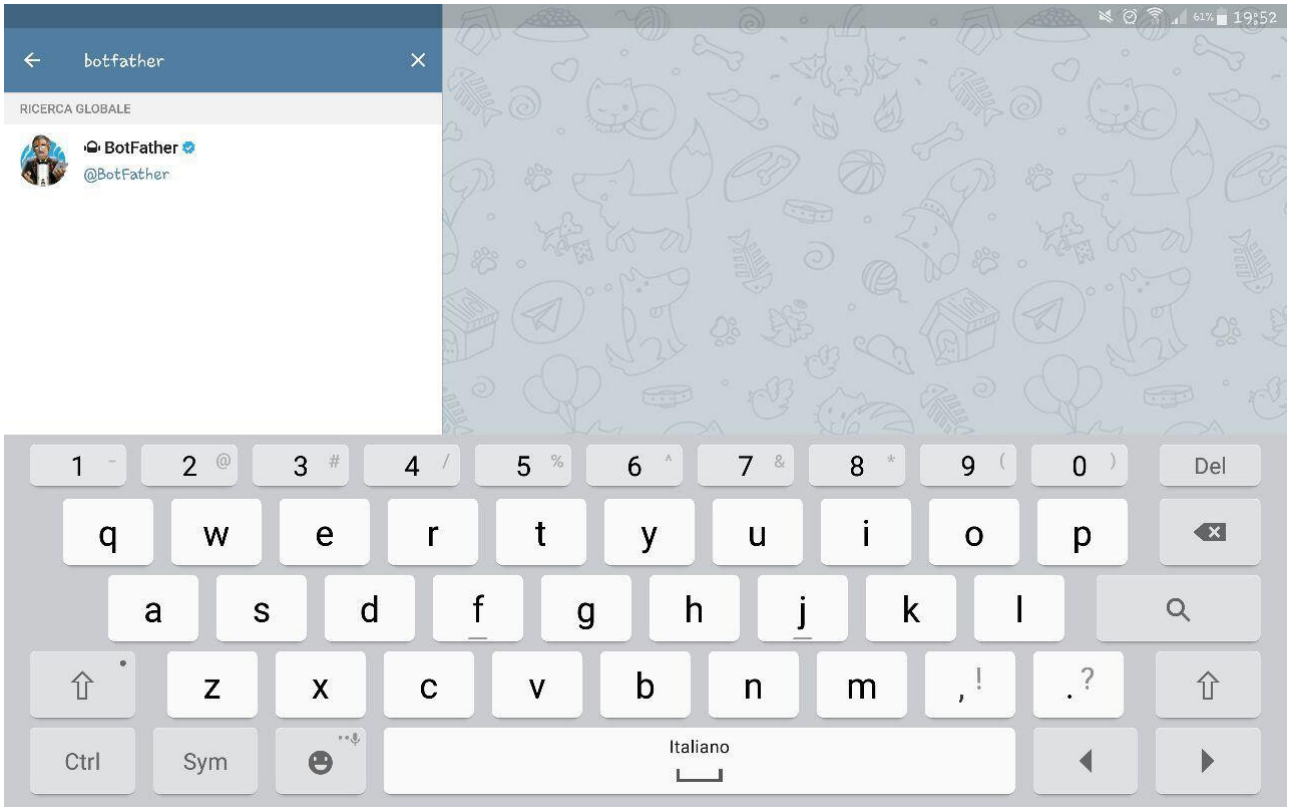
Fishgram is an interesting library that allows the management of our Fishino via a simple message of Telegram. In this pdf file we will find a full explanation of the example Receive_Send_Message.

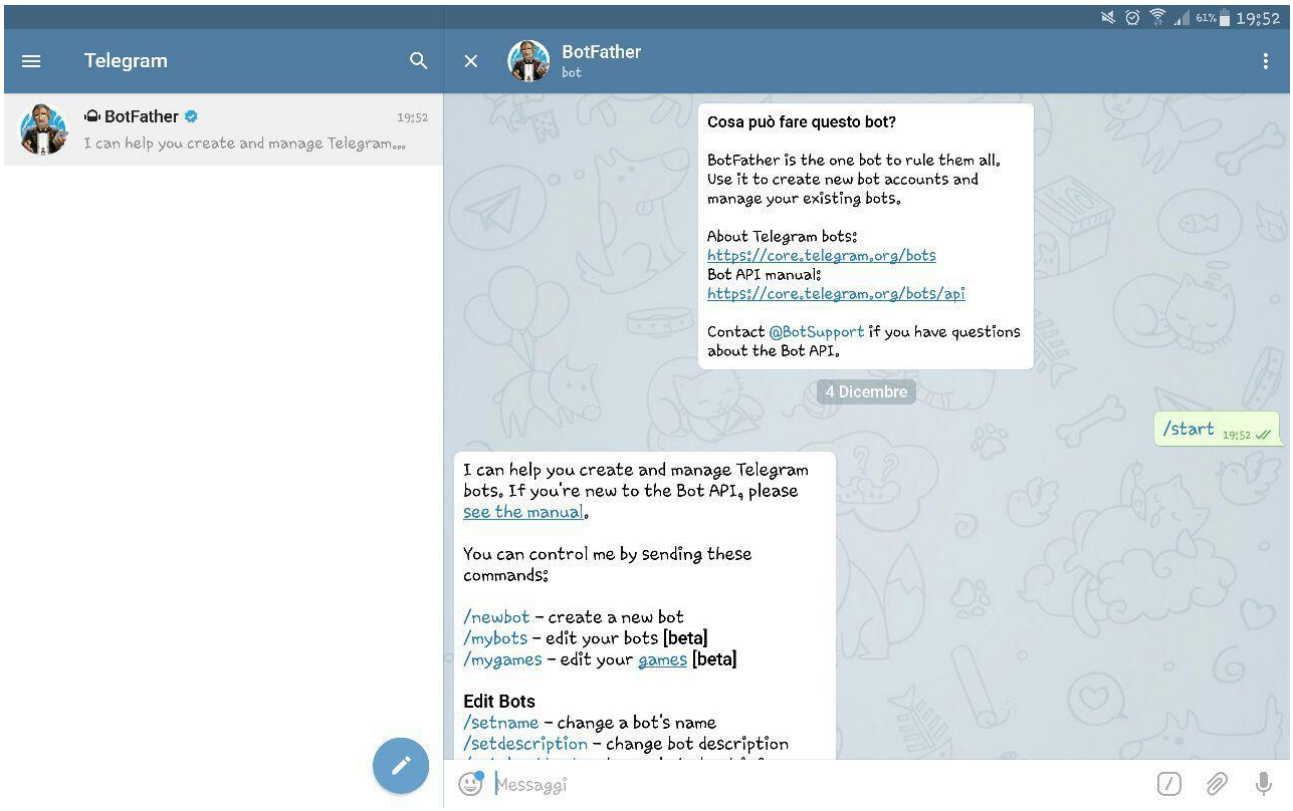
First we have to download Telegram for free on our mobile device (Android - iOS - Windows - Linux - ...) by connecting on the store or on https://telegram.org/.



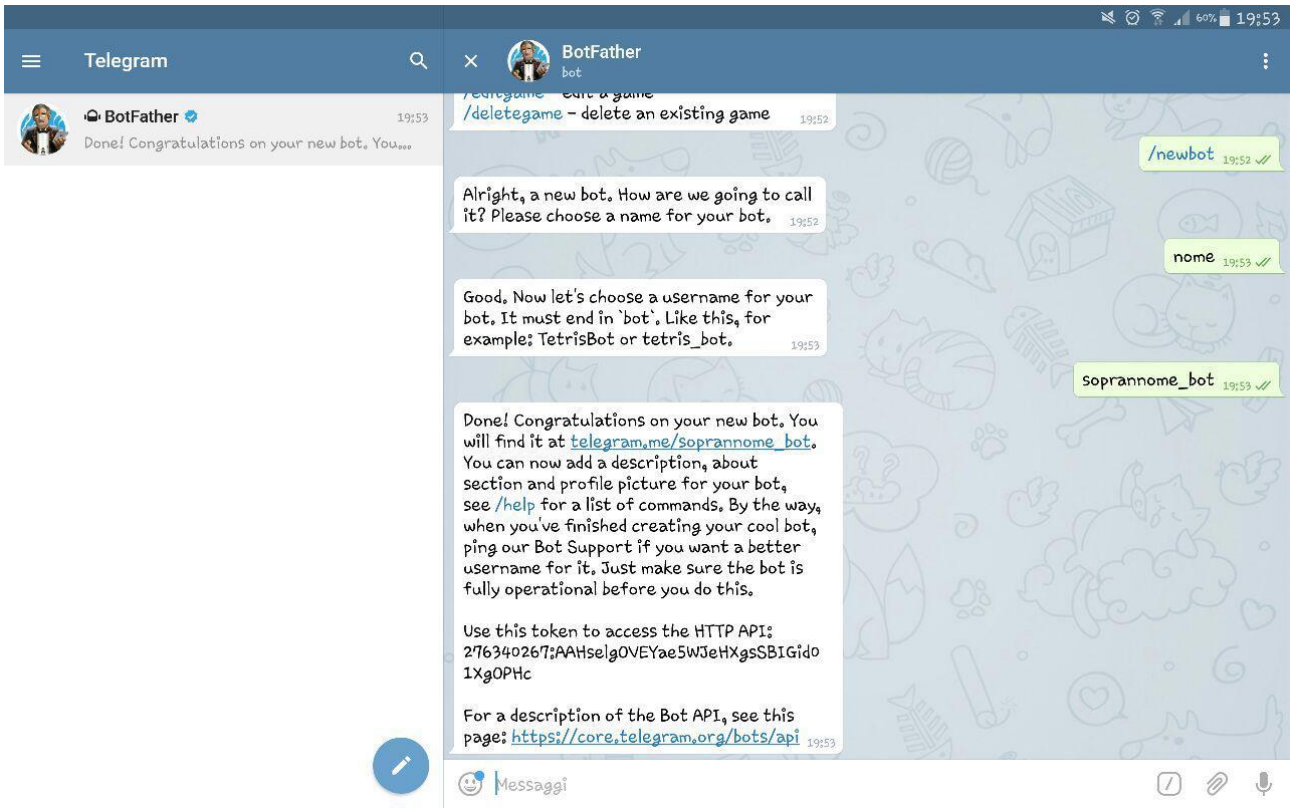
After registration, using the search tool, we must find Botfather and start it to create our bots.





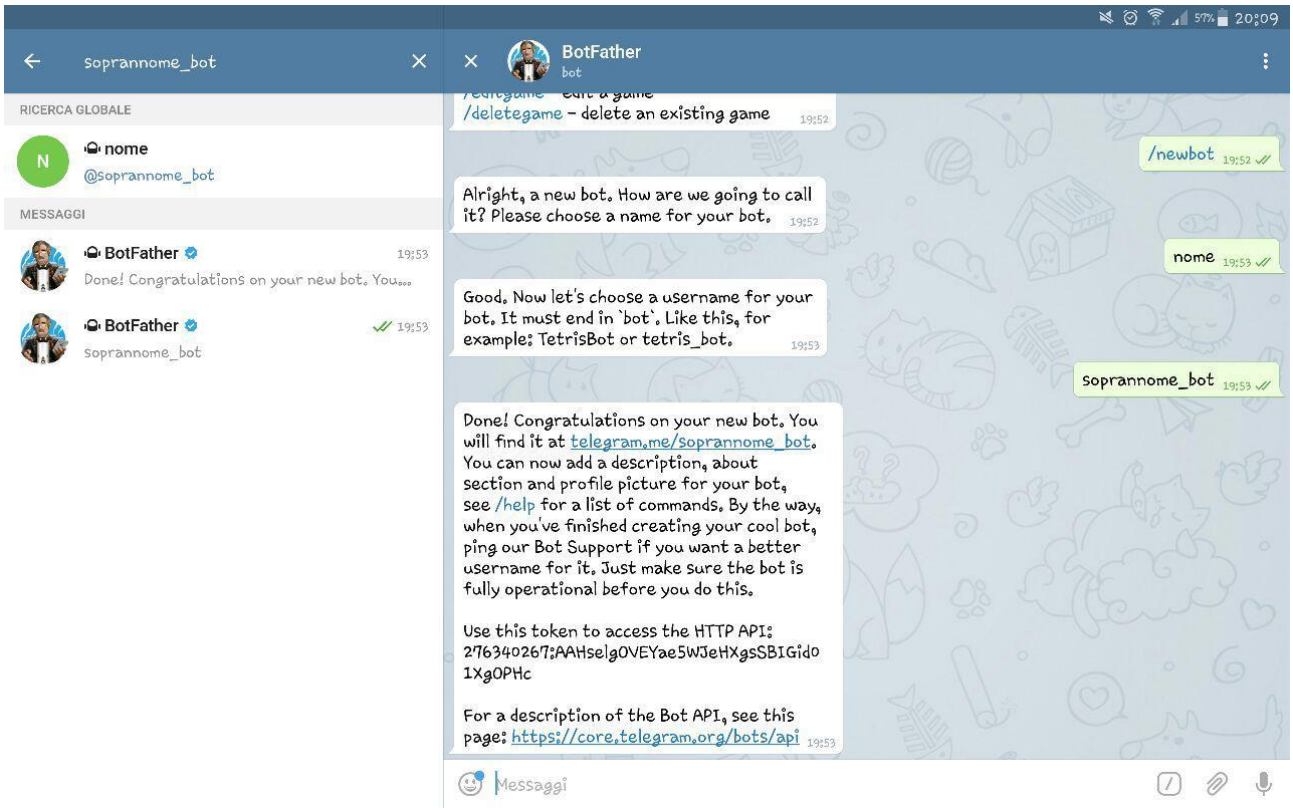


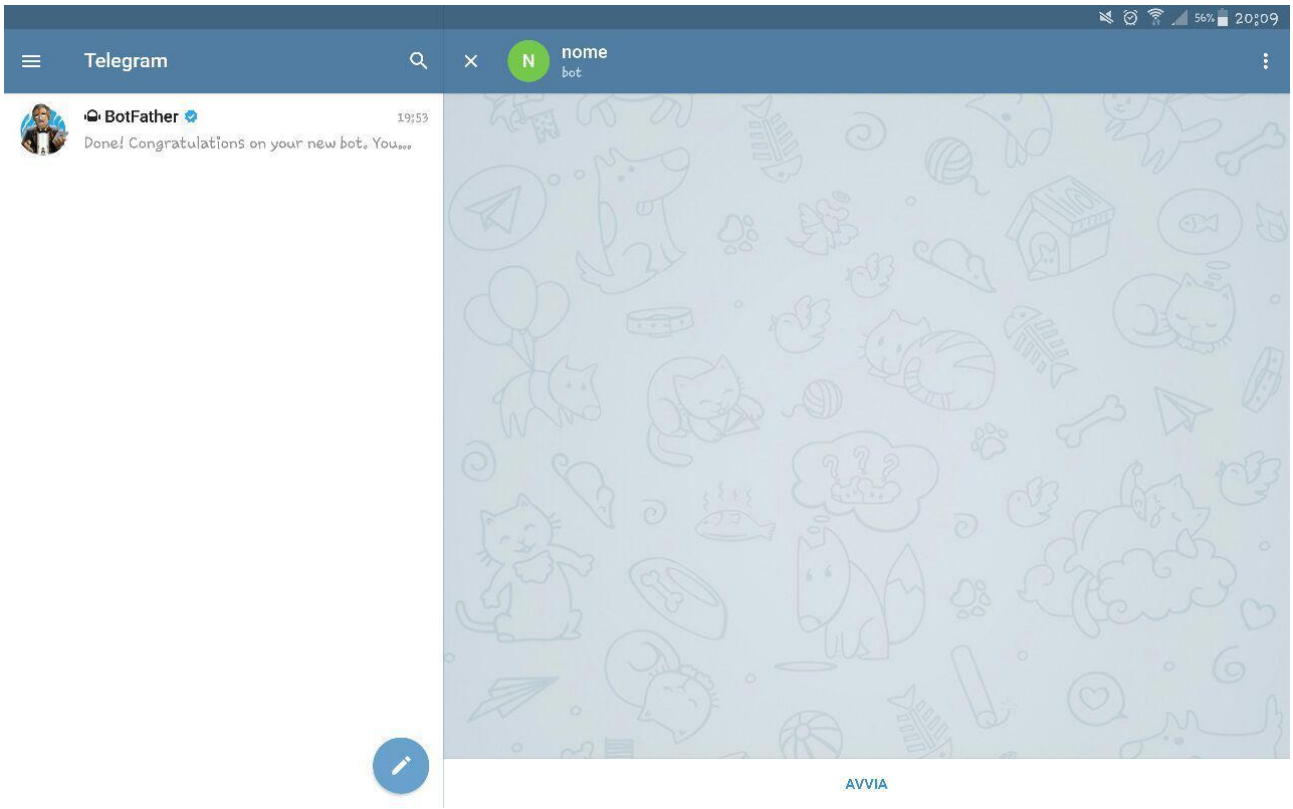
The process is very simple:
- Press /newbot
- Choose a name
- Choose a nickname ending in "bot"

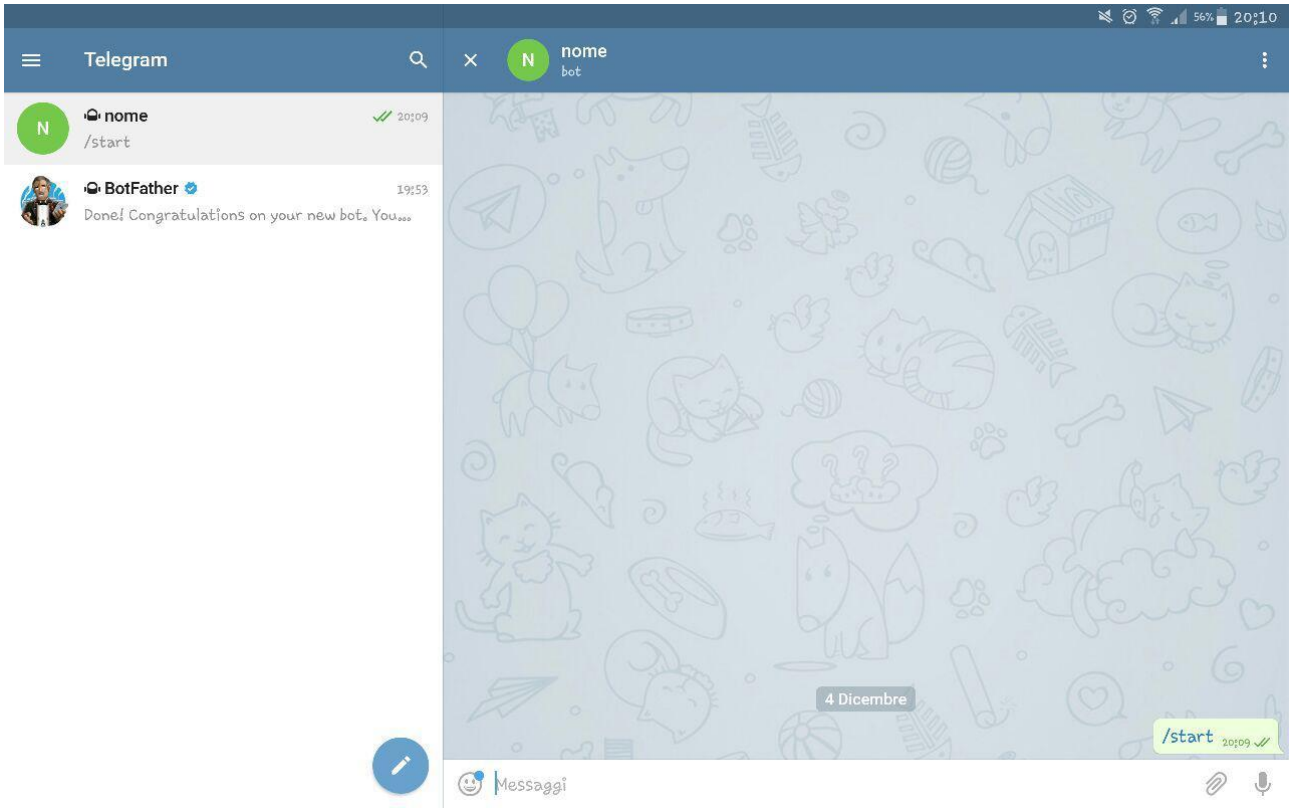


We will need the token, so let's write it somewhere:
276340267:AAHselgOVEYae5WJeHXgsSBIGid01XgOPHc

Now we must allow the bot to communicate with us by activating it.
In the search box we must find our bot by searching through the nickname that we have chosen. Open the chat and press START.


soprannome_bot
RICERCA GLOBALE
nome
@soprannome_bot
MESSAGGI
BotFather
19:53
Done! Congratulations on your new bot. You...
BotFather
19:53
soprannome_bot
BotFather
bot
/deletegame – delete an existing game 19:52
/newbot 19:52
Alright, a new bot. How are we going to call it? Please choose a name for your bot. 19:52
nome 19:53
Good. Now let's choose a username for your bot. It must end in `bot`. Like this, for example: TetrisBot or tetris_bot. 19:53
soprannome_bot 19:53
Done! Congratulations on your new bot. You will find it at telegram.me/soprannome_bot. You can now add a description, about section and profile picture for your bot, see /help for a list of commands. By the way, when you've finished creating your cool bot, ping our Bot Support if you want a better username for it. Just make sure the bot is fully operational before you do this.
Use this token to access the HTTP API:
276340267:AAHselgOVEYae5WJeHXgsSBIGid0
1Xg0PHc
For a description of the Bot API, see this page: https://core.telegram.org/bots/api 19:53
Messaggi


Telegram
BotFather
19:53
Done! Congratulations on your new bot. You...
nome
bot
AVVIA



Now we will be able to communicate with the bot and it will be able to communicate with us.
It is noteworthy that to turn on our bot was enough to look for and press start, therefore it means that anyone can do it. This is true. But remember that only who created the bot can edit its properties through BotFather or know the token. In addition, each message sent to the Bot is marked by a user id. So you can put a check in the sketch to perform only authorized id's instructions.
The remaining messages will be read but ignored.

Let's study the sketch.

After including the necessary libraries and after setting SSID, password, etc to connect to a WiFi network we find:

```
FishinoSecureClient client;

const char accessToken[] = "*********:**********************************";
```

The first line of code creates a client object that supports secure connections (https). In the second one we need to enter the token that we have just found. In our case the second line will become:

```
const char accessToken[] = "276340267:AAHselgOVEYae5WJeHXgsSBIGid01XgOPHc ";
```

Only after entering the latter two lines of code we insert:

```
Fishgram bot(accessToken, client);
```

This instruction creates an instance of the Fishgram class and requires two parameters: the token in the form of const char and the client object that we created. In this case the instance name is "bot".
Then we find the only authorized id, although we are not in any way limited to having just one.

```
long id_john = ********;
```

To know our id simply open a browser and enter the following string (making sure to place your token) and press enter:
https://api.telegram.org/access_token/getUpdates

My string will be:
https://api.telegram.org/bot304107554:AAHpD4LXvsGid9mQLMpQlqEcMqLA7iDa5TE/getUpdates

In response you should see a string similar to this:

{"ok":true,"result":[{"update_id":638542571,
"message":{"message_id":2,"from":{"id":********,"first_name":"Andrea Simone","last_name":"Costa"},"chat":{"id":°°°°°°°°,"first_name":"Andrea Simone","last_name":"Costa","type":"private"},"date":1480927961,"text":"/start","entities":[{"type":"bot_command","offset":0,"length":6}]}}]}

The 8 digits covered by the asterisks * are your id.

If for some reason you displayed a string like this:
{"Ok": true, "result": []}
may you have not started the bot from your mobile device.
Start it, or write anything and press enter. Then try refreshing your browser page.

It's interesting to see how the Telegram bot does not have access to our phone number. When we register to Telegram, after entering our name, we gets a unique id that will uniquely identify us in the universe of Telegram. This id is visible to the bot.

Now we can analyze the void setup and void loop of our code, then understand the use of two external procedures.

The only relevant statement in the setup is:

```
resetSetESP();
```

This procedure manages the settings of the ESP module and allows, after an initial reset, to connect to our wifi network. Later we will see why these instructions are not present only in the setup, so we will understand why it is necessary to include them in an external void.

In the loop we find a 2000 ms delay to avoid sending requests to the server Telegram too frequently.
After calling timeEsp (), the second external procedure, we find:

```
oldestMessage receivedMessage = bot.getOldestMessage();
```

With this instruction we create a oldestMessage variable called receivedMessage. This is nothing more than a simple structure used internally by Fishgram library, in which we find all the necessary information of the more dated message, but not yet analyzed, that a user has written to our bot using Telegram.
The call to getOldestMessage method returns a oldestMessage object, that we store in the receivedMessage variable .

A oldestMessage structure has four data members:

-isEmpty is a Boolean variable that is true if the JSON message received by Telegram is empty or is an invalid string, false otherwise. In other words, Fishino repeatedly perform requests to Telegram but most of the answers it receives will not contain any messages. A check using this variable is useful to act only in the case we have received something.

-sender_id is a string with the sender's id.

-chat_id is a string with the chat's id.

-text is a string containing the received text. Any message sent to our bot is marked with a initial quotation mark " and a final quotation mark ". For example, if we want using the command pin3On active Fishino pin 3, we can test the equality between the text string and a string of our choice. As mentioned above the latter string will not be pin3On, but will be "pin3On".

Continuing in the loop we find:

```
unsigned long senderID = (receivedMessage.sender_id).toInt()
```

This statement creates an unsigned long variable called senderID, where we store the result of sender's id conversion from string to integer. This id, using getOldestUpdates, is returned as a string (`receivedMessage.sender_id` is a string) and we convert it to unsigned long to be able to compare with another long variable. Which? One that contains the authorized id.

Now we find an if construct, where there are two conditions:

- `(!receivedMessage.isEmpty)`

we ensure that the JSON message from Telegram contains a non-empty and valid message.

- `(senderID == id_john)`

Finally we make sure that the sender's id is authorized.

If these conditions are met we print the received message's text::

```
Serial.println(receivedMessage.text);
```

We create a string to be sent as a reply:

```
const char reply [] = " Hi, how are you?";
```

We reply to the message received using sendMessage method:

```
bot.sendMessage(reply, receivedMessage.sender_id);

bot.sendMessage(receivedMessage.text, receivedMessage.sender_id);
```

The required parameters for this method are the text to be sent in the form of const char or string, and the id of the recipient. In this example we reply to the user that sent a message with some text and then with the same text received earlier.

Because the code in the library is not yet fully stable could happen that Fishino blocks. The cause of this block still resides in a lack of communication with Telegram, so you only need to restart the ESP module. This is why the instructions to reset and set it are inserted in `resetSetESP();`, to be called when needed.
It's convenient set a reset at periodic intervals through the millis () function. This reset is run by the procedure `timeEsp()`, so don't remove it and don't change it.
The only helpful change might be modify the time interval between two reset. By default it's set to 600,000 milliseconds (ten minutes) using `interval` variable.