**TELEGRAM BOT**

[**https://t.me/BotFather**](https://t.me/BotFather)

**Hay que interactuar con él, para poder crear un Bot**

* **Dentro de telegram**

**/newbot**

* **Elige un nombre para tu Bot**

**thor\_timmy94\_bot**

* **Creamos dentro de /03\_server una carpeta llamada /telegramBot**
* **Dentro, en la Terminal**

**Solo genera el package.json**

**npm init -y**

* **Instalación de librerías**

**npm install express dotenv telegraf**

* **Creamos a la altura del proyecto el fichero /.gitignore**

/node\_modules

.env

* **Dentro de package.json y configuramos los scripts**

  "scripts": {

    "start": "node index.js",

    "dev": "nodemon index.js"

},

* **Creamos a la altura del proyecto creamos el fichero index.js**

**Lo primero que necesitamos es traernos express pues la requerimos como siempre.**

const express = *require*('express');

**generamos una aplicación, es la que no da la posibilidad de crear rutas…**

const app = *express*();

**Una vez teniendo la aplicación de express generada la ponemos a escuchar por el puerto por defecto o por el archivo .env**

const PORT = process.env.PORT || 3000;

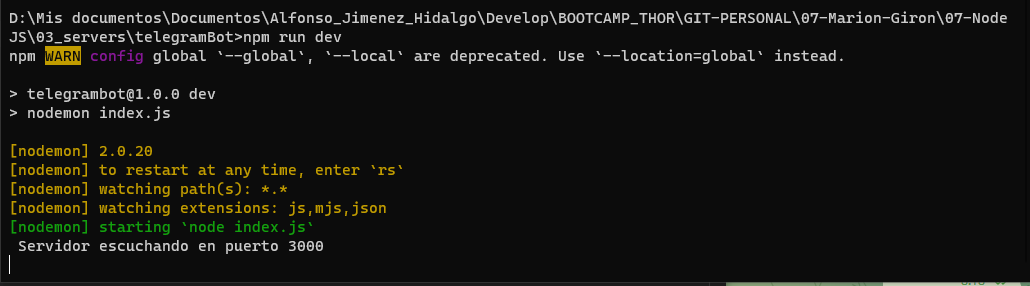
app.*listen*(PORT, () => {

    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

**Probamos desde la Terminal**

**npm run dev**



**Vamos adelantarnos para hacer la carga del fichero de entorno también dentro de index.js, recordad para poder enlazar el fichero para ello hemos instalado primero la librería dotenv**

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

**A la altura del proyecto creamos el fichero de entorno .env**

PORT=3000

**Creamos una ruta de tipo post para que el Bot nos pueda enviar las cosas, dentro de index.js**

router.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

**index.jshasta ahora.**

const express = *require*('express');

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

const app = *express*();

app.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

const PORT = process.env.PORT || 3000;

app.*listen*(PORT, () => {

    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

**Dentro de index.js vamos a requerir una librería telegraf pero solo vamos a decirle que solo lo que tenga la librería telegraf requerimos solo telegraf.**

const { Telegraf } = *require*('telegraf')

**Creamos una variable como un nuevo bot**

const bot = new *Telegraf*(proces.env.BOT\_TOKEN);

**Vamos al fichero de entorno .env y le pasamos el token que nos ha pasado cuando hemos creado el bot en telegram.**

BOT\_TOKEN="5722384637:AAE4grKnbjBhjN98RUfGWDhHTP2WPV6lWIk"

**Lo único que nos queda ahora es la configuración del Bot**

**Dentro de index.js**

**Necesitamos una url para poder comunicarnos con el**

**Le decimos que todo lo que entre por la url url\_telegram pásalo por aquí**

app.*use*(bot.*webhookCallback*('/url\_telegram'));

**Cuál es la url que va estar colocada online para que el servidor de telegram pueda mandarnos cosas**

bot.telegram.*setWebhook*(`${process.env.BOT\_URL} /url\_telegram`)

**Para que nos sea más fácil abrimos otro Terminal a parte y levantamos ngrok**

**ngrok http 3000**

[**https://fb61-2a0c-5a82-2302-eb00-28a4-8863-c915-70ca.eu.ngrok.io**](https://fb61-2a0c-5a82-2302-eb00-28a4-8863-c915-70ca.eu.ngrok.io)

**Dentro del fichero de entorno .env**

BOT\_URL="https://fb61-2a0c-5a82-2302-eb00-28a4-8863-c915-70ca.eu.ngrok.io"

**NOTA :**

**PUES YA TENEMOS LA ESTRUCTURA BÁSICA PAR APODER COMUNICARNOS CON TELGRAM ES DECIR CON NUESTRO BOT**

**COMANDOS BOT**

**Dentro de index.js creamos los comandos**

**Toda la información la tenemos en el context (ctx)**

*//COMANDOS*

bot.*command*('test', (ctx) => {

    ctx.*reply*('Encantado de conocerte');

})

**Index.js**

const express = *require*('express');

const { Telegraf } = *require*('telegraf')

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

const app = *express*();

const bot = new *Telegraf*(process.env.BOT\_TOKEN);

*//Configuración de BOT*

app.*use*(bot.*webhookCallback*('/url\_telegram'));

bot.telegram.*setWebhook*(`${process.env.BOT\_URL}/url\_telegram`)

app.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

*//COMANDOS*

bot.*command*('test', (ctx) => {

    ctx.*reply*('Encantado de conocerte');

})

const PORT = process.env.PORT || 3000;

app.*listen*(PORT, () => {

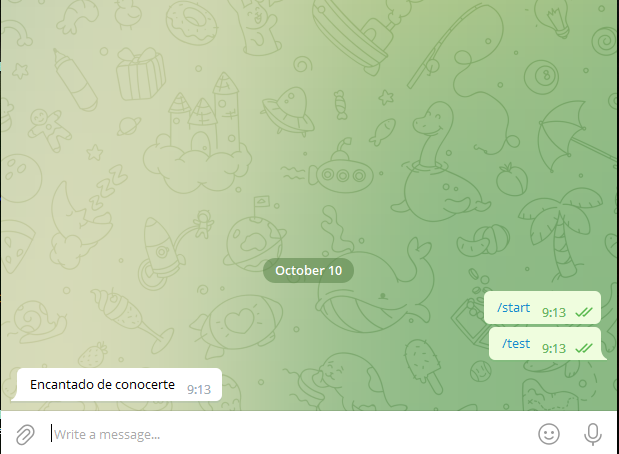
    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

**Si todo ha ido pues en la url que da bot father le hacemos click y probamos el comando que hemos creado**

**t.me/thor\_timmy94\_bot**

**/test**



Dentro de index.js le hacemos un console.log para ver los datos y asi saber que captar más adelante.

*//COMANDOS*

bot.*command*('test', (ctx) => {

    console.*log*(ctx.message);

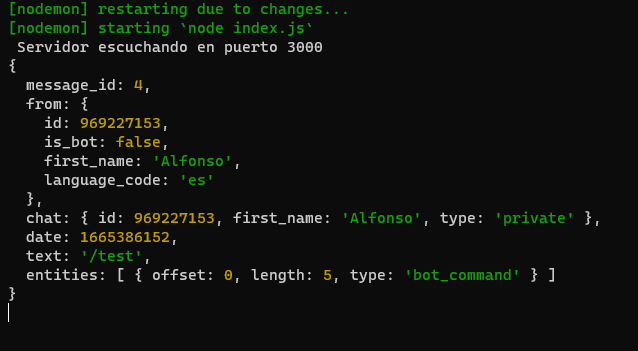
    ctx.*reply*('Encantado de conocerte');

})

Lanzamos en telegram Alfonsito

/test

Y nos vamos a la Terminal para ver que nos aparece



Todo el comando que creamos para lanzar los comandos lo más cómodo es llevarla a otro fichero

A la altura del proyecto creamos la carpeta /commads y dentro un fichero llamado test.js

Pues cogemos la función del index y la pegamos en test.js

Lo mismo que ponemos en el /model es lo mismo que recibimos en el index.js

module.exports = *async* (ctx) => {

    console.*log*(ctx.message);

*await* ctx.*replyWithDice*();

*await* ctx.*reply*('Encantado de conocerte');

}

NOTA: await nos respeta el orden .

Dentro de index.js requerimos lo que tenemos en text.js

*//COMANDOS*

bot.*command*('test', *require*('./commands/test'));

index.js

const express = *require*('express');

const { Telegraf } = *require*('telegraf')

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

const app = *express*();

const bot = new *Telegraf*(process.env.BOT\_TOKEN);

*//Configuración de BOT*

app.*use*(bot.*webhookCallback*('/url\_telegram'));

bot.telegram.*setWebhook*(`${process.env.BOT\_URL}/url\_telegram`)

app.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

*//COMANDOS*

bot.*command*('test', *require*('./commands/test'));

const PORT = process.env.PORT || 3000;

app.*listen*(PORT, () => {

    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

NOTA: ESTO FUNCIONA IGUAL, ES PURA ORGANIZACIÓN PARA UN FUTURO Y NO DEJAR TODO EL CODIGO EN INDEX.JS DE LOS COMANDOS CREADOS.

EJERCICIO PROFE

# Comando /tiempo

- OBJETIVO: **Recuperar ciertos datos de temperaturas de una ciudad seleccionada a partir de la ejecución de un comando en el Bot.**

-EJECUCION

**-/tiempo Madrid**

**-/tiempo Santiago de Compostela**

**-/tiempo Berlín**

# TAREAS:

**-Definir en index.js y en su fichero correspondiente el comando /tiempo**

-¡Probar que funciona!

-Extraer la ciudad seleccionada

**- ¿De dónde saco el texto? --> Hay que mirar ctx.message**

**- De ese mensaje, ¿Cómo extraigo únicamente la ciudad?**

- ¿Cómo saco los valores de temperatura de esa ciudad?

**-API: https://openweathermap.org/**

**-Hay que registrarse para obtener un API Key**

**-Mediante la URL: https://api.openweathermap.org/data/2.5/weather?q={cityname}&appid={APIkey}&units=metric**

**-Para recuperar la información a partir de esa URL usamos la librería**

**AXIOS (GET sobre la URL anterior)**

-Como respuesta enviaremos los datos de:

**Temperatura actual, Temperatura Max, Temperatura Min, Humedad**

Dentro de index.js

bot.*command*('tiempo', *require*('./commands/tiempo'));

Creamos dentro de la carpeta /commands el fichero tiempo.js

module.exports = (ctx) => {

    ctx.*reply*('Comando tiempo');

}

Extraemos la ciudad dentro de tiempo.js

const axios = *require*("axios");

module.exports = *async* (ctx) => {

    const ciudad = ctx.message.text.*split*('/tiempo')[1];

    const url = `https://api.openweathermap.org/data/2.5/weather?q=${ciudad}&appid=${process.env.OWM\_API\_KEY}&units=metric`;

    try {

        const { data } = *await* axios.*get*(url);

        const respuesta = `El tiempo en ${ciudad}:

🌡Actual: ${data.main.temp}º

🔥 Máxima: ${data.main.temp\_max}º

🥶 Mínima: ${data.main.temp\_min}º

💧 Humedad: ${data.main.humidity}%`;

        ctx.*reply*(respuesta)

    } catch (err) {

        ctx.*reply*('Ciudad no encontrada ❌');

    }

}

Creamos en el fichero de entorno .env

PORT=3000

BOT\_TOKEN="5722384637:AAE4grKnbjBhjN98RUfGWDhHTP2WPV6lWIk"

BOT\_URL="https://2629-2a0c-5a82-2302-eb00-28a4-8863-c915-70ca.eu.ngrok.io"

OWM\_API\_KEY="04f4f8762b4ce35df1c074d18b1e84ab"

Probamos a lanzar en telegram /tiempo Granada



EJERCICIO PROFE

##**Comando /donde**

**-OBJETIVO: A partir de una dirección devolvernos la posición exacta.**

**-EJECUCION:**

**-/donde Calle Gran Vía 23, Madrid**

**###TAREAS:**

**-Generar el fichero que gestiona el comando.**

**-Extraer la dirección del texto**

**-La librería node-geocoder nos permite recuperar lat y lng a partid de una dirección.**

**- Con ctx.replyWithLocation ( lat, lng) podemos responder con un mapa interactivo**

**Creamos un nuevo comando en index.js**

*//Donde*

bot.*command*('donde', *require*('./commands/donde'));

**Creamos un fichero dentro de commands llamado donde.js**

const NodeGeocoder = *require*('node-geocoder');

const options = {

    provider: 'google',

    apiKey: process.env.GOOGLE\_API\_KEY,

};

module.exports = *async* (ctx) => {

    const direccion = ctx.message.text.*split*('/donde')[1];

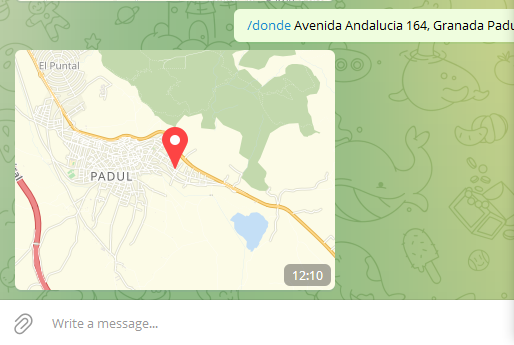
    const geocoder = *NodeGeocoder*(options);

    const res = *await* geocoder.*geocode*(direccion);

    ctx.*replyWithLocation*(res[0].latitude, res[0].longitude);

}

**Probamos en telegram y pasamos dirección**



**BOT RESPONDE CON TEXTO**

**Los hacemos a través de eventos.**

**Dentro de index.js**

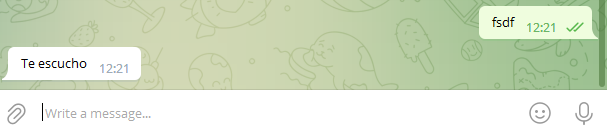
*//eventos Bot responde cont texto*

bot.*on*('text', (ctx) => {

    ctx.*reply*('Te escucho');

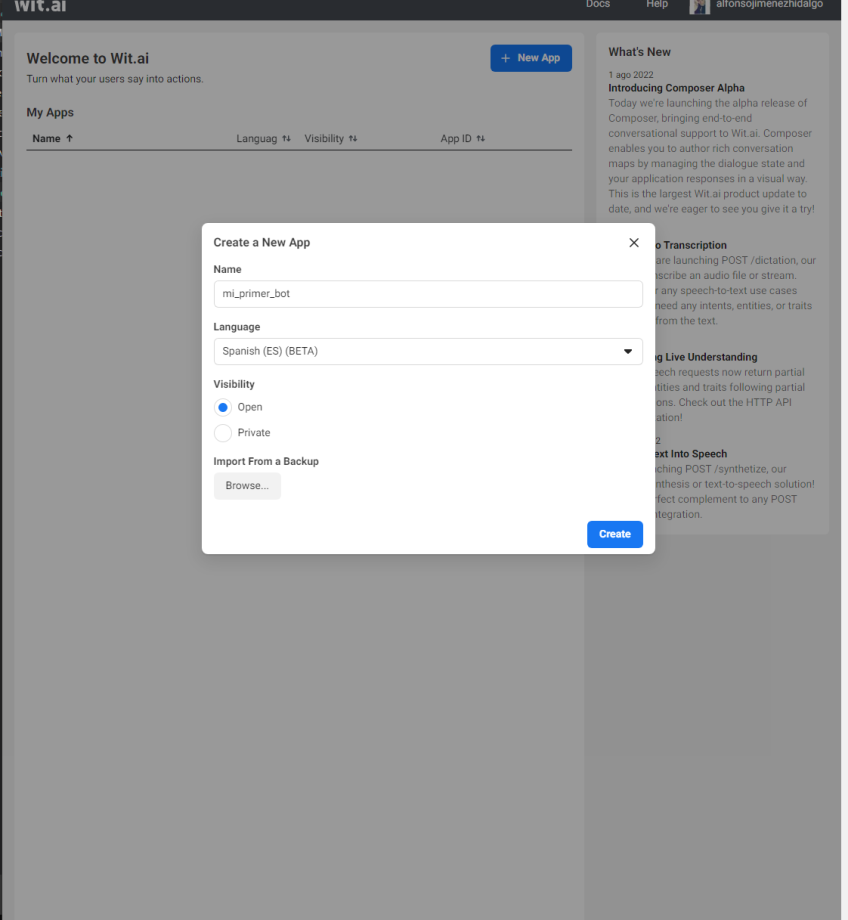
})

**Cualquier cosa que le escribamos al bot en telegram nos va a responder te escucho**

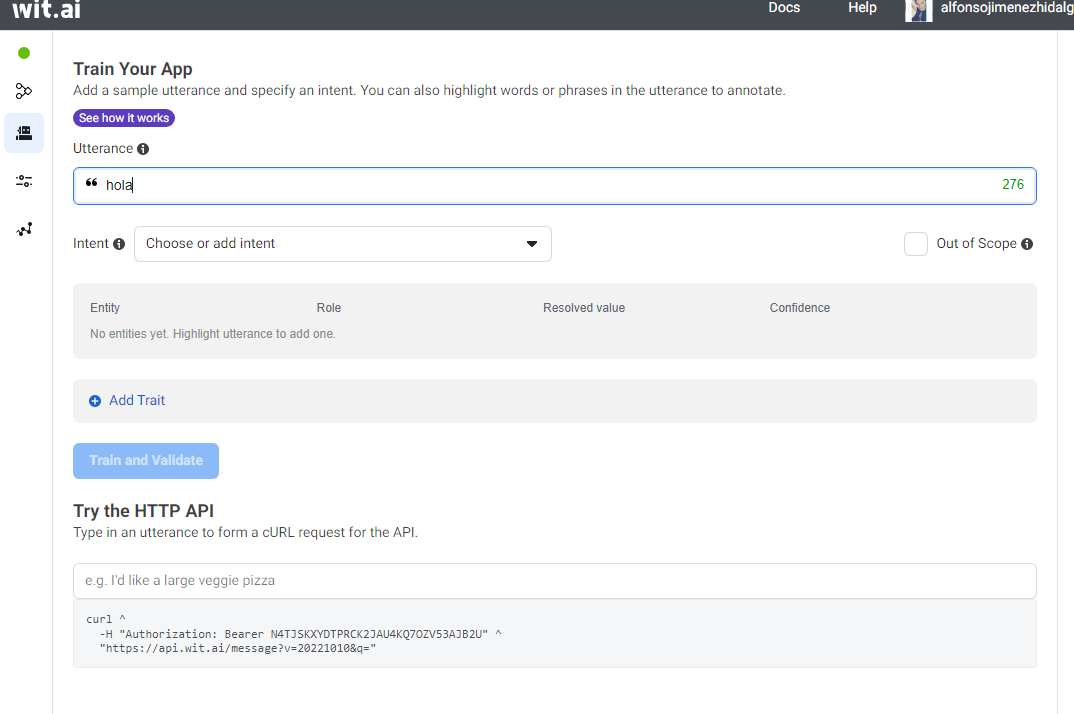


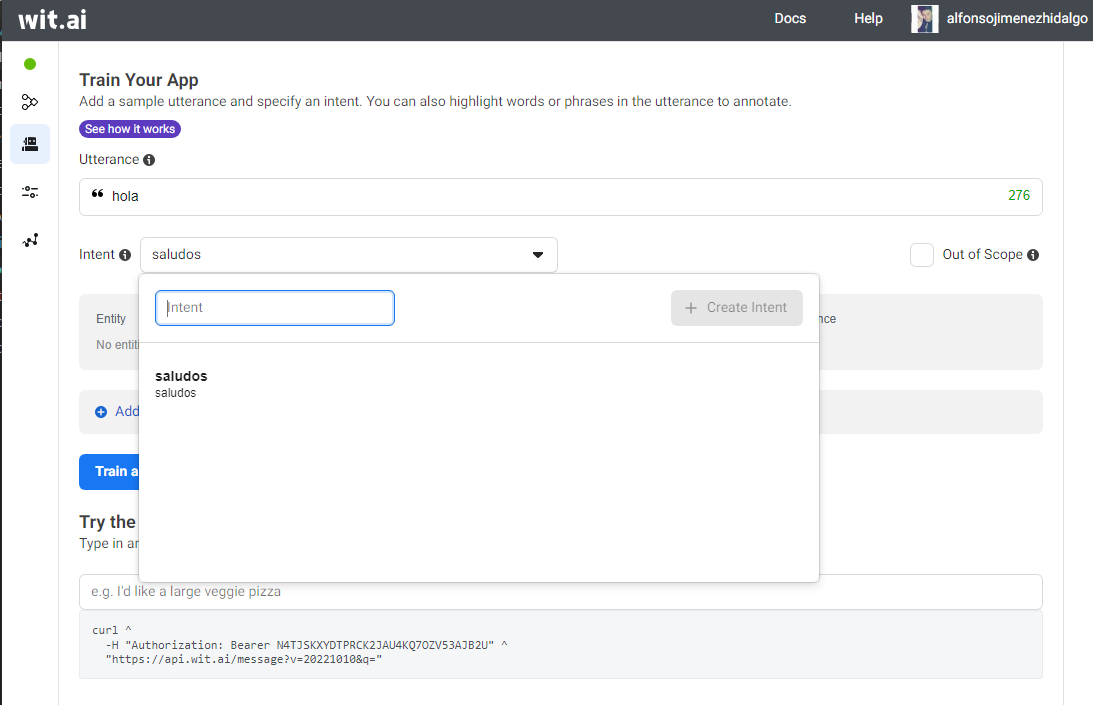
**Vamos a enseñarle cosas a nuestro BOT.**

[**https://wit.ai/apps**](https://wit.ai/apps)

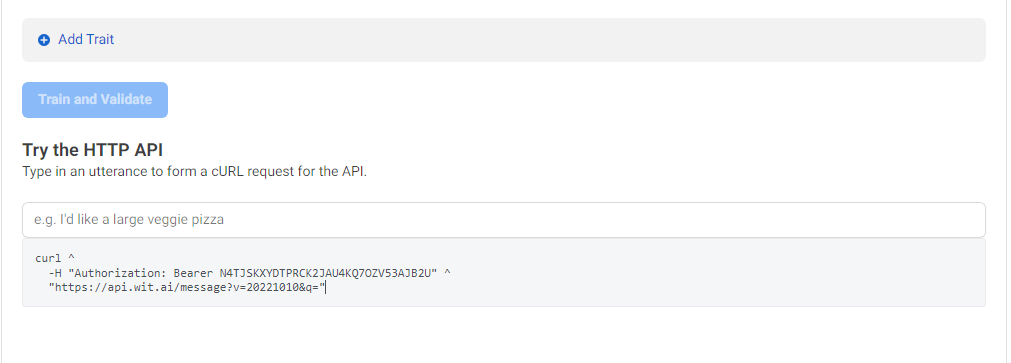


**Aquí le podemos entrenarlo.**

**Y le ponemos que cuando digan hola que sepa que es un saludo**



**Le damos a Train and Validate para que vaya a entrenarlo.**

**.. Dejamos que termine de entrenar.**

A la altura del proyecto creamos un nuevo fichero nlu.js

Instalamos la librería npm node-wit

const { Wit, log } = *require*("node-wit")

module.exports = *async* (ctx) => {

    const client = new *Wit*({

        accessToken: process.env.WIT\_TOKEN,

        logger: new log.*Logger*(log.DEBUG)

    });

    const response = *await* client.*message*(ctx.message.text);

    console.*log*(response);

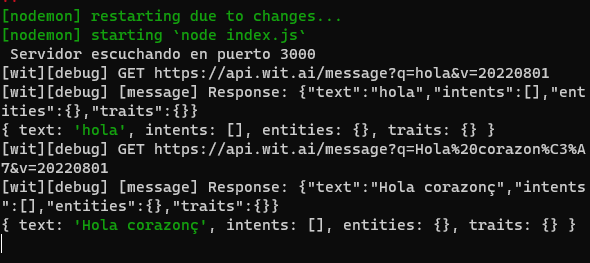
    ctx.*reply*('Te escucho');

}

**Dentro del index.js**

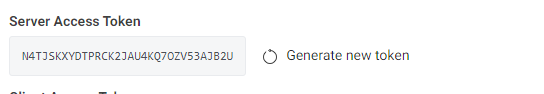
*//eventos Bot responde cont texto*

bot.*on*('text', *require*('./nlu'));

**Si lanzamos en telegram cualquier mensaje por consola no debería de responder con un objeto con hola**

**Dentro de .env le pasamos el token de la web wit.ai**

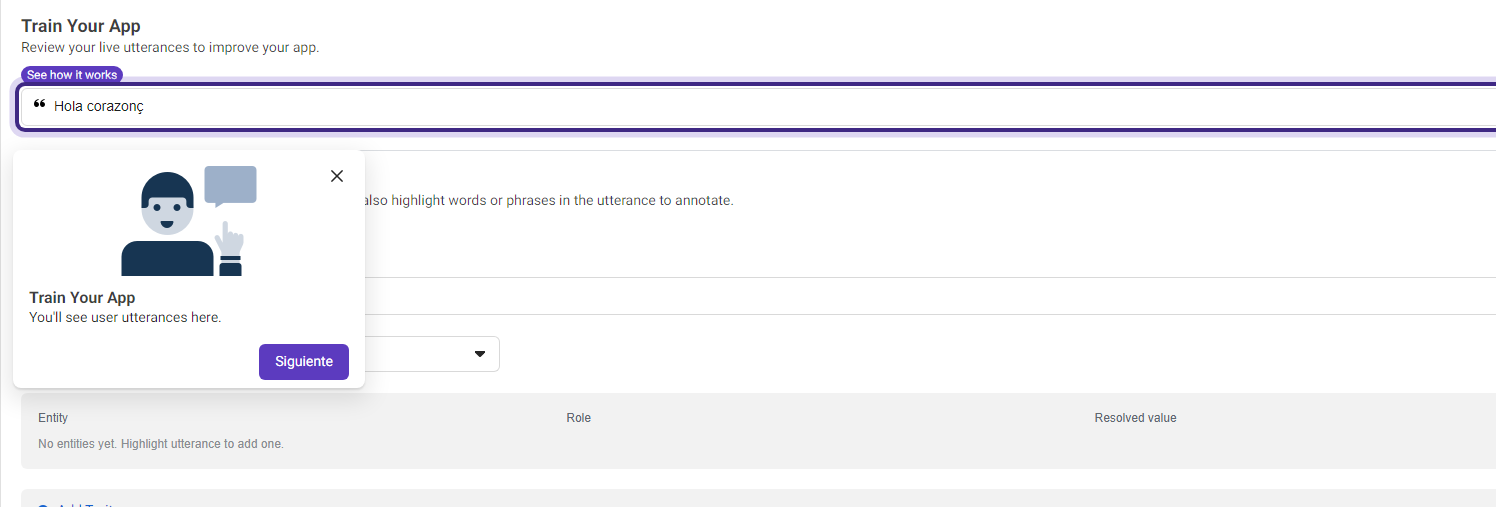
**Setting / Server token**



.env

WIT\_TOKEN="N4TJSKXYDTPRCK2JAU4KQ7OZV53AJB2U"

**En la web de wit.ai debería aparecer lo que escribimos en telegram en Understanding**



**Para ver que ha aprendido lo que hemos creado con hola**

const { Wit, log } = *require*("node-wit")

module.exports = *async* (ctx) => {

    const client = new *Wit*({

        accessToken: process.env.WIT\_TOKEN,

        logger: new log.*Logger*(log.DEBUG)

    });

    const response = *await* client.*message*(ctx.message.text);

*if* (response.intents.length === 0) {

        ctx.*reply*('No te entiendo');

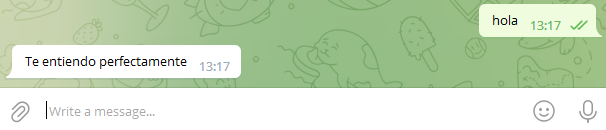
    } *else* {

        ctx.*reply*('Te entiendo perfectamente')

    }

}

**Lo probamos en telegram**



**Vamos a crear una serie de ficheros para que nos responda con una frase.**

**Creamos una carpeta phrases y dentro un fichero de saludo.js y otro de despedida.js**

**Dentro de cada uno de ellos creamos varias frases y diferentes líneas para más adelante poder elegir aleatoriamente una de ellas.**

**Dentro de nlu.js**

const { Wit, log } = *require*("node-wit")

const fs = *require*('fs');

module.exports = *async* (ctx) => {

    const client = new *Wit*({

        accessToken: process.env.WIT\_TOKEN,

        logger: new log.*Logger*(log.DEBUG)

    });

    const response = *await* client.*message*(ctx.message.text);

*if* (response.intents.length === 0) {

        ctx.*reply*('No te entiendo');

    } *else* {

        const intent = response.intents[0].name;

*//1- Rellenar los ficheros con saludos y despedidas (uno por fila)*

*//2- A partir del intent, leer el fichero correspondiente.*

*//3- Coger una frase aleatoria del fichero (split ('\n))*

*//4- Responder con dicha frase.*

        const content = fs.*readFileSync*(`./phrases/${intent}.txt`, 'utf-8');

        const frases = content.*split*('\n');

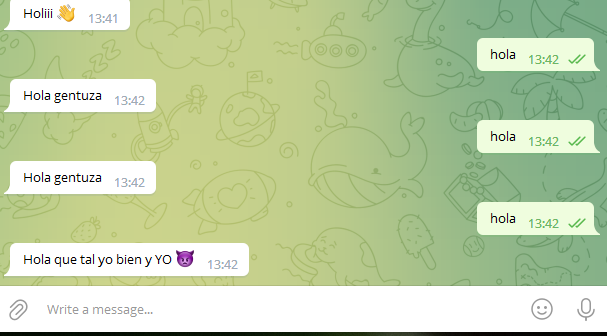
        const randowNum = Math.*floor*(Math.*random*() \* frases.length);

        ctx.*reply*(frases[randowNum]);

    }

}

**Probamos por telegram**



**Ahora vamos a hacer que el Bot nos responda hablando.**

**Con la librería tts-api**

**Terminal**

**npm install google-tts-api**

**dentro de nlu.js**

**importamos es decir requerimos**

const { Wit, log } = *require*("node-wit")

const fs = *require*('fs');

const googleTTS = *require*('google-tts-api');

module.exports = *async* (ctx) => {

    const client = new *Wit*({

        accessToken: process.env.WIT\_TOKEN,

        logger: new log.*Logger*(log.DEBUG)

    });

    const response = *await* client.*message*(ctx.message.text);

*if* (response.intents.length === 0) {

        ctx.*reply*('No te entiendo');

    } *else* {

        const intent = response.intents[0].name;

*//1- Rellenar los ficheros con saludos y despedidas (uno por fila)*

*//2- A partir del intent, leer el fichero correspondiente.*

*//3- Coger una frase aleatoria del fichero (split ('\n))*

*//4- Responder con dicha frase.*

        const content = fs.*readFileSync*(`./phrases/${intent}.txt`, 'utf-8');

        const frases = content.*split*('\n');

        const randowNum = Math.*floor*(Math.*random*() \* frases.length);

        const url = googleTTS.*getAudioUrl*(frases[randowNum], {

            lang: 'es',

            slow: *false*

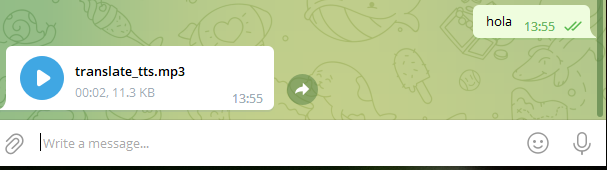
        });

        ctx.*replyWithAudio*(url);

    }

}

**Lo probamos por telegram**



**Vamos a integrar una base de datos, para poder registrar usuarios…**

**Instalmos librería**

**npm install mysql2**

**Configurar la base de datos en un fichero de entorno .env**

PORT=3000

BOT\_TOKEN="5722384637:AAE4grKnbjBhjN98RUfGWDhHTP2WPV6lWIk"

BOT\_URL="https://2629-2a0c-5a82-2302-eb00-28a4-8863-c915-70ca.eu.ngrok.io"

OWM\_API\_KEY="04f4f8762b4ce35df1c074d18b1e84ab"

GOOGLE\_API\_KEY="AIzaSyBMOcTcAkobrlfKIBOJNz6lDw2R5fJsk\_Q"

WIT\_TOKEN="25RJY3UPBNTJSNIYXUF4VIKREAXO5KPY"

DB\_HOST="127.0.0.1"

DB\_USER="root"

DB\_PASSWORD="1804017"

DB\_PORT=3306

DB\_NAME="telegramBot"

**Creamos fichero db.js**

const mysql = *require*('mysql2');

const pool = mysql.*createPool*({

    host: process.env.DB\_HOST,

    user: process.env.DB\_USER,

    password: process.env.DB\_PASSWORD,

    port: process.env.DB\_PORT,

    database: process.env.DB\_NAME

});

global.db = pool;

**Creamos el fichero utils.js**

const *executeQuery* = (query, params = []) => {

*return* new Promise((resolve, reject) => {

        db.*query*(query, params, (err, result) => {

*if* (err) *reject*(err);

*resolve*(result);

        });

    });

}

*/\*\**

*\* Ejecuta una sentencia SQL y devuelve el primer elemento recogido en la query o el valor null si no encuentra dicho elemento*

*\* @param {\*} query sentencia SQL a ejecutar*

*\* @param {\*} params array con los valores a sustituir dentro de la sentencia*

*\* @returns una promesa con el resultado de la ejecución de la query*

*\*/*

const *executeQueryOne* = (query, params = []) => {

*return* new Promise((resolve, reject) => {

        db.*query*(query, params, (err, result) => {

*if* (err) *return* *reject*(err);

*if* (result.length === 0) *resolve*(null);

*resolve*(result[0]);

        });

    });

}

module.exports = {

    executeQuery, executeQueryOne

}

**Dentro de index.js**

**Cargamos la base de datos.**

**Para que se conecte.**

const express = *require*('express');

const { Telegraf } = *require*('telegraf')

const axios = *require*('axios');

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

*//Carga de la base de datos*

*require*('./config/db');

const app = *express*();

const bot = new *Telegraf*(process.env.BOT\_TOKEN);

*//Configuración de BOT*

app.*use*(bot.*webhookCallback*('/url\_telegram'));

bot.telegram.*setWebhook*(`${process.env.BOT\_URL}/url\_telegram`)

app.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

*//COMANDOS*

*//test*

bot.*command*('test', *require*('./commands/test'));

*//tiempo*

bot.*command*('tiempo', *require*('./commands/tiempo'));

*//Donde*

bot.*command*('donde', *require*('./commands/donde'));

*//eventos Bot responde cont texto*

bot.*on*('text', *require*('./nlu'));

const PORT = process.env.PORT || 3000;

app.*listen*(PORT, () => {

    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

**Yo quiero guardar el usuario que esté interactuando con mi Bot.**

**También podemos crear MIDDLEWARE.**

**Dentro de index.js**

*//MIDDLEWARE*

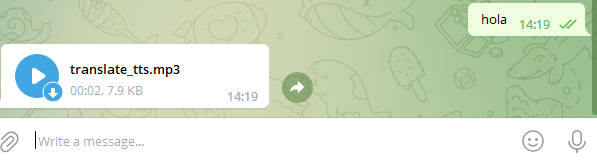
bot.*use*((ctx, next) => {

    console.*log*(ctx.from);

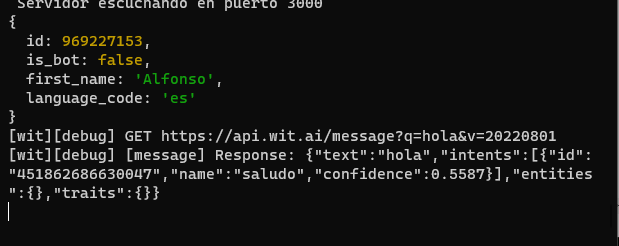
*next*();

});

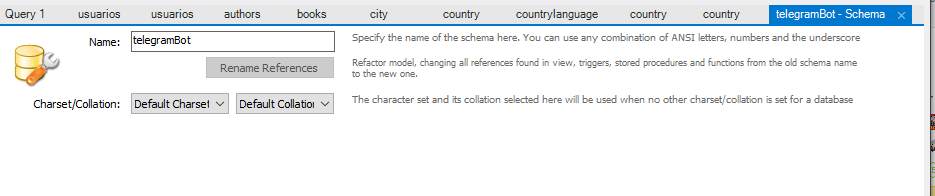
**Por telegram**



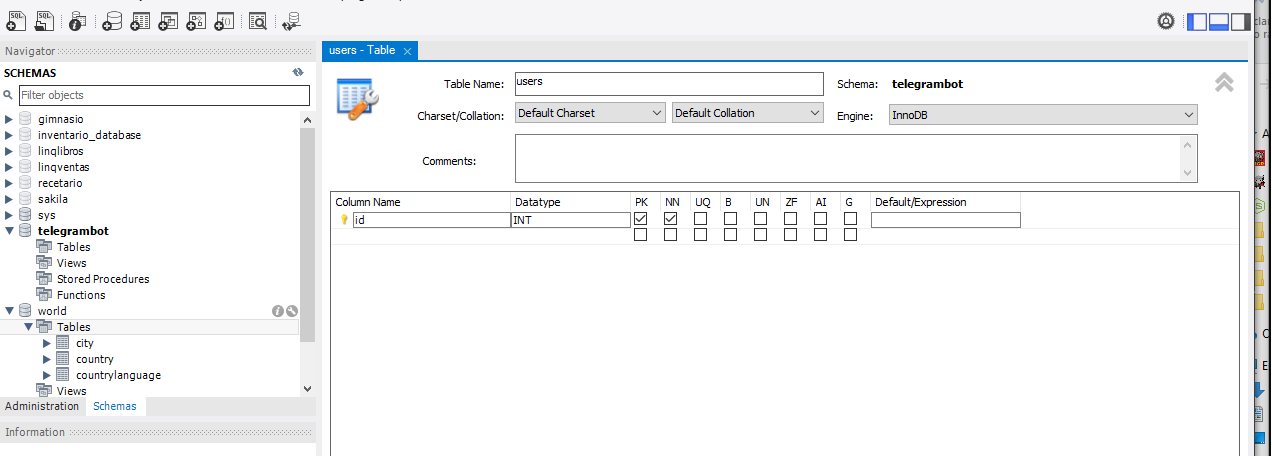
Y por consola nos aparecerá el usuario que ha mandado el mensaje



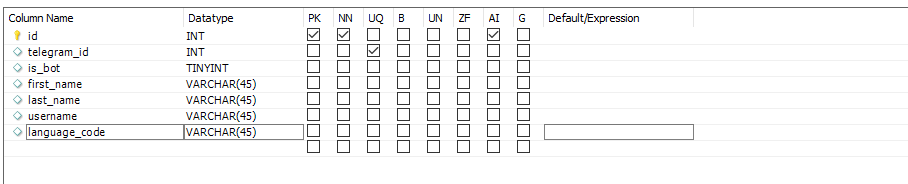
**Ahora vamos a crear un nuevo esquema en Workbench**



**Creación de tabla.**



**Creamos los campos**



**Creamos una nueva carpeta models**

**Dentro de models creamos el fichero user.model.js**

**Dentro de user.model.js**

const { executeQuery } = *require*("../helpers/utils");

const *create* = ({id, is\_bot, first\_name, last\_name, username, language\_code}) => {

*return* *executeQuery*('insert into users (telegram\_id, is\_bot, first\_name, last\_name, username, language\_code) values (?, ?, ?, ?, ?, ?)', [id, is\_bot, first\_name, last\_name, username, language\_code]);

}

module.exports = {

    create

};

**Dentro de index.js en Middleware**

const express = *require*('express');

const { Telegraf } = *require*('telegraf')

const axios = *require*('axios');

const { create } = *require*('./models/user.model');

*//Carga del fichero de entorno*

*require*('dotenv').*config*();

*//Carga de la base de datos*

*require*('./config/db');

const app = *express*();

const bot = new *Telegraf*(process.env.BOT\_TOKEN);

*//Configuración de BOT*

app.*use*(bot.*webhookCallback*('/url\_telegram'));

bot.telegram.*setWebhook*(`${process.env.BOT\_URL}/url\_telegram`)

app.*post*('/url/telegram', (req, res) => {

    res.*send*('Termina');

});

*//MIDDLEWARE*

bot.*use*(*async* (ctx, next) => {

    console.*log*(ctx.from);

    try {

*await* *create*(ctx.from);

    } catch (err) {

        console.*log*('No se inserta el usuario');

    } finally {

*next*();

    }

});

*//COMANDOS*

*//test*

bot.*command*('test', *require*('./commands/test'));

*//tiempo*

bot.*command*('tiempo', *require*('./commands/tiempo'));

*//Donde*

bot.*command*('donde', *require*('./commands/donde'));

*//eventos Bot responde cont texto*

bot.*on*('text', *require*('./nlu'));

const PORT = process.env.PORT || 3000;

app.*listen*(PORT, () => {

    console.*log*(` Servidor escuchando en puerto ${PORT} `);

});

**Pues listo cuando escribamos la primera vez en el chat en telegram directamente nos inserta en la BD de MySQL los datos del usuario.**

