

Laboratorio: Creando un bot usando Microsoft Bot Framework en Node.js

Introducción

Objetivos

Después de completar estos laboratorios, serás capaz de tener una comprensión de los conceptos básicos Microsoft Bot Framework como herramienta para optimizar la construcción de bots.

Requisitos previos

- Node.js
- Línea de comandos
- Comprensión básica de Node.js y JavaScript

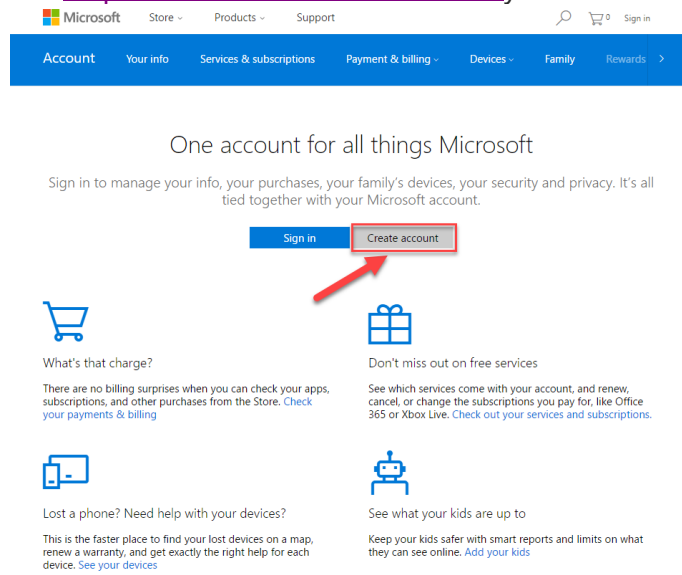
Instalación y configuración

1. Instalar el software prerequisite

- **Node.js:** <https://NodeJS.org/en/> (esto incluye npm)
- **NGrok:** <https://ngrok.com/>
- **Skype:** <http://Skype.com> (si desea probar un Skype)
- **Visual Studio Code:** <https://code.VisualStudio.com/>

2. Crear un ID de Microsoft (si no tienes uno)

- Ir a <https://account.Microsoft.com/> y crear una cuenta.



- En el cuadro Nombre de usuario introduzca su dirección de correo electrónico existentes y haga clic en obtener una nueva dirección de correo electrónico para crear una dirección de Hotmail o Outlook.

Microsoft account opens a world of benefits.

☒ Send me promotional emails from Microsoft

[Use a phone number instead](#) **OR** [Get a new email address](#)

Choosing Next means that you agree to the [Microsoft Services Agreement](#) and [privacy and cookies statement](#).

Next

Microsoft

NOTA: Si utilizas una dirección de correo electrónico existente necesitas verificar antes de avanzar.

Verify email

We sent an email to danielegan0112@gmail.com to make sure you own it. Please check your inbox and follow the instructions to finish setting up your Microsoft account.

[Use a different email address as your Microsoft account](#)

Resend email

[Terms of Use](#) [Privacy & Cookies](#) [Sign out](#)

Microsoft

Google

Gmail

COMPOSE

Inbox (1)
Starred
Sent Mail
Drafts
More

Daniel

Verify your email address

Microsoft account team <account> to me (x) 10:20 AM (2 minutes ago)

Microsoft account

Verify your email address

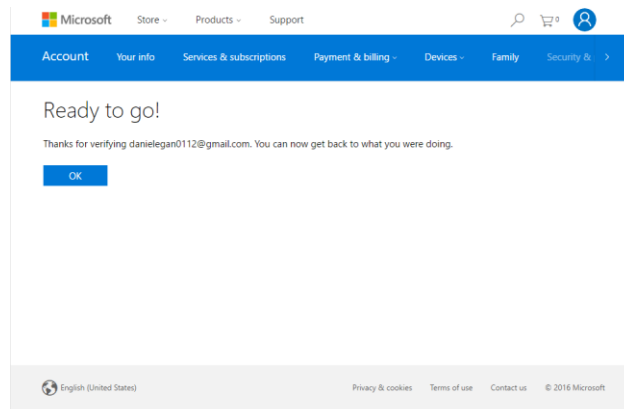
To finish setting up this Microsoft account, we just need to make sure this email address is yours.

Verify danielegan0112@gmail.com

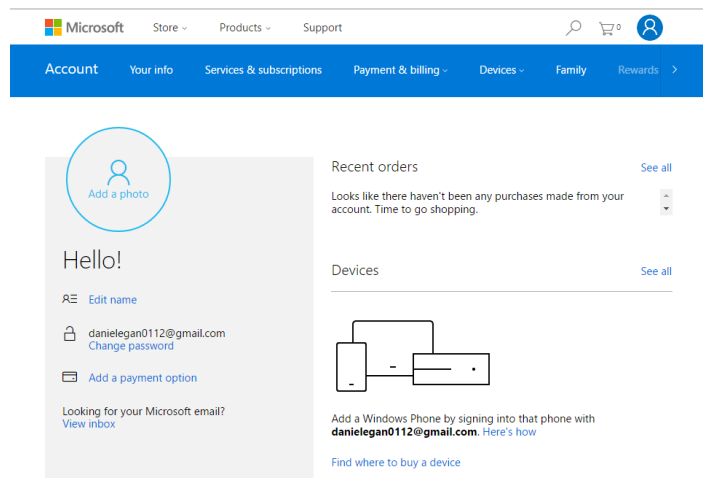
Or you may be asked to enter this security code: 9862

If you didn't make this request, [click here](#) to cancel.

Thanks,
The Microsoft account team

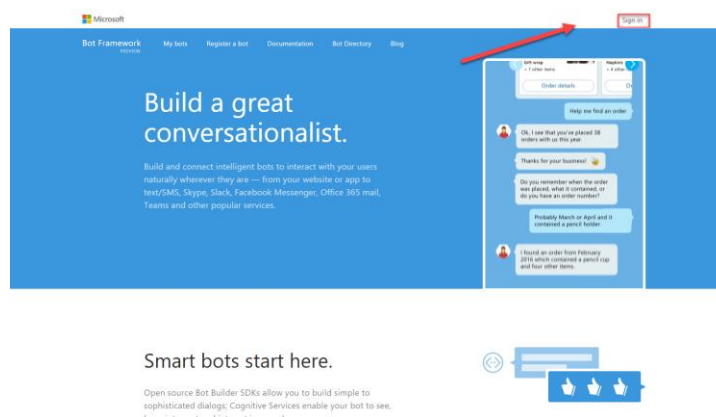


- Cualquier camino te llevará a esta pantalla

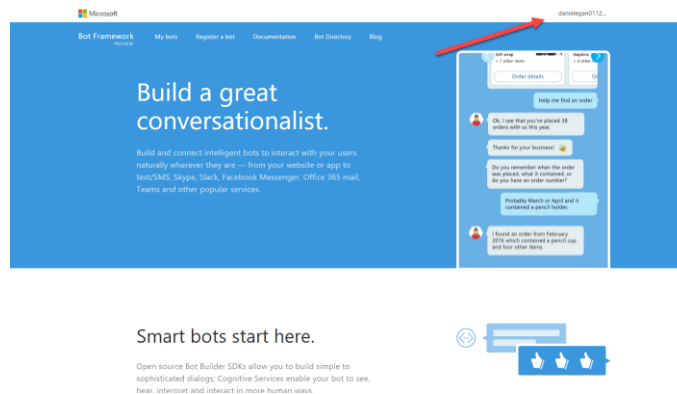


3. Crear una cuenta de Bot Framework

- Desplácese hasta <http://BotFramework.com>
- Haga clic en "Sign In"



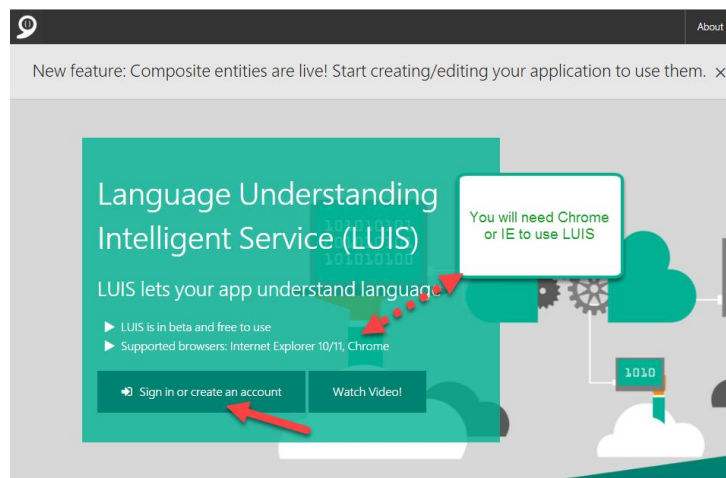
- Si está usando el mismo navegador que usó para crear su ID de Microsoft iniciará sesión automáticamente, de lo contrario necesitará utilizar el ID que acaba de crear para iniciar sesión en.



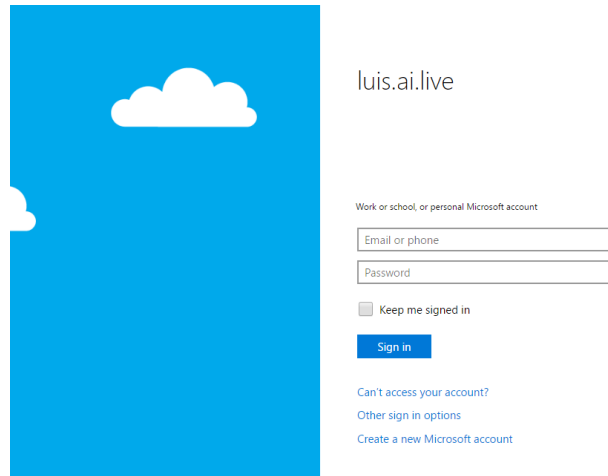
Puede dejar esta ventana abierta; la usaremos luego.

4. Registrarse para LUIS, “Servicios Inteligentes para la Comprensión de Lenguaje”, <https://www.luis.ai>

- Haga clic en Iniciar Sesión, o Crear Cuenta



- Inicia sesión con tu cuenta de Microsoft



luis.ai.live

Work or school, or personal Microsoft account

Email or phone

Password

☐ Keep me signed in

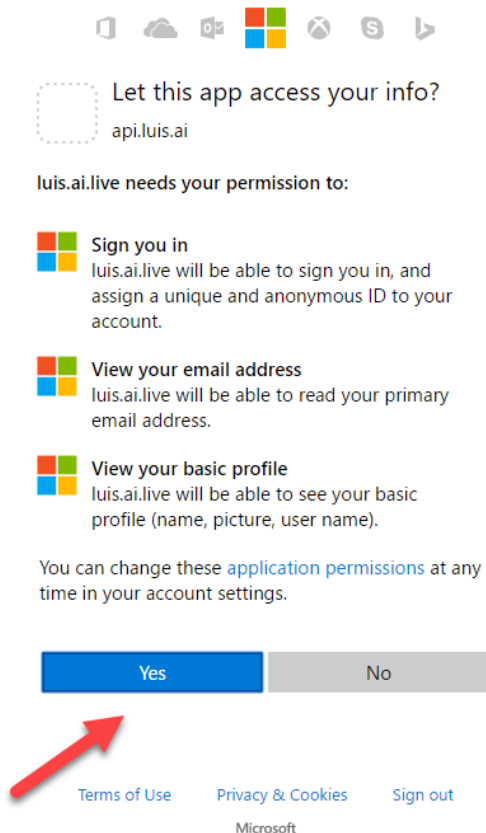
Sign in

[Can't access your account?](#)

[Other sign in options](#)

[Create a new Microsoft account](#)


- Si usted todavía está firmado en le pedirá que diga sí para aceptar los permisos. De lo contrario tendrá que iniciar sesión con el ID de Microsoft que creó anteriormente.





Let this app access your info?

api.luis.ai

luis.ai.live needs your permission to:

 **Sign you in**
luis.ai.live will be able to sign you in, and assign a unique and anonymous ID to your account.

 **View your email address**
luis.ai.live will be able to read your primary email address.

 **View your basic profile**
luis.ai.live will be able to see your basic profile (name, picture, user name).

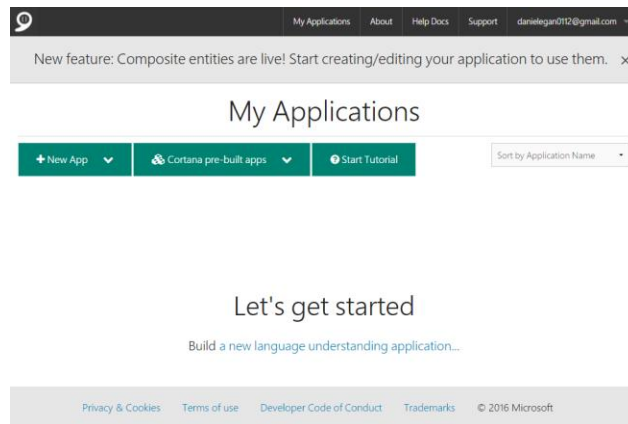
You can change these [application permissions](#) at any time in your account settings.

Yes No

[Terms of Use](#) [Privacy & Cookies](#) [Sign out](#)

Microsoft

- Usted puede caminar a través del tutorial rápido si desea o ignórelo. Cuando haya terminado, su pantalla debe parecerse a la siguiente:



Copiar/pegar del código

Usted tendrá la opción de copiar y pegar fragmentos de código de este documento para completar este laboratorio. Aprenderás mucho más escribiéndolo, pero para agilizar el desarrollo del lab, se brinda la opción anterior.

Nota : Si estás en un Mac, utiliza el archivo PDF. No copies y pegues desde el archivo PDF. Hay un archivo independiente denominado SNIPS.txt que contiene los recortes que necesita.

Ejercicio 1: Bot básica usando BotBuilder

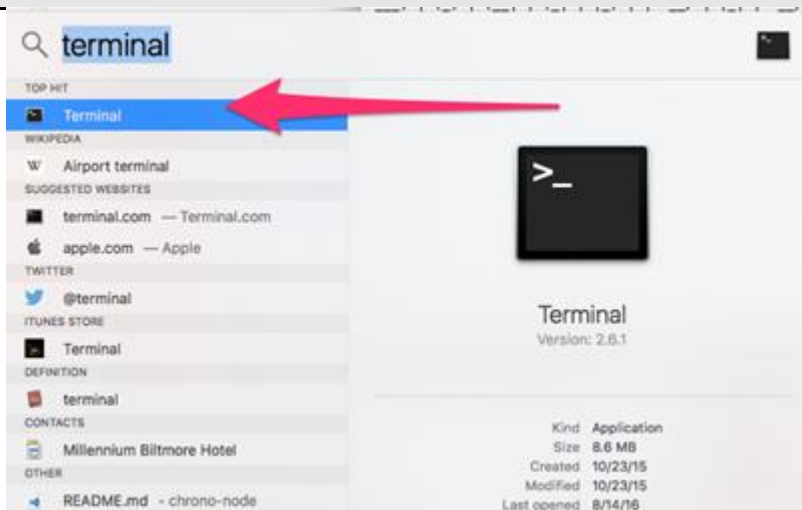
En este ejercicio, va a crear un bot sencillo usando el BotBuilder

Pasos detallados

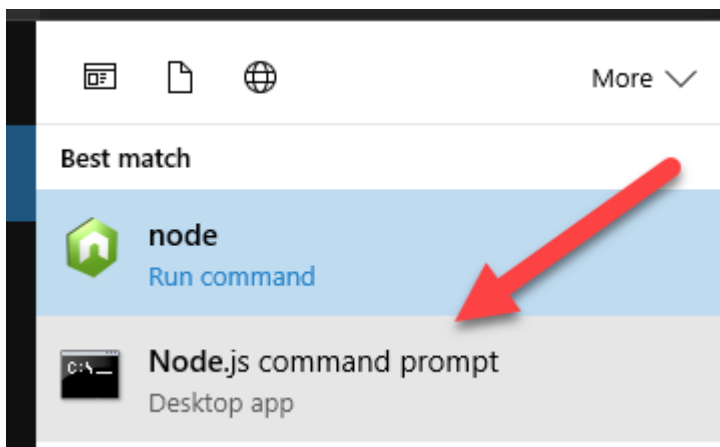
1. Abrir el Terminal. (Esto podría ser Terminal en Mac, PowerShell en Windows, o una terminal de su elección (cmdr, iTerm, etcetera...))

En Mac: **comando** → **Barra espaciadora** y terminal tipo

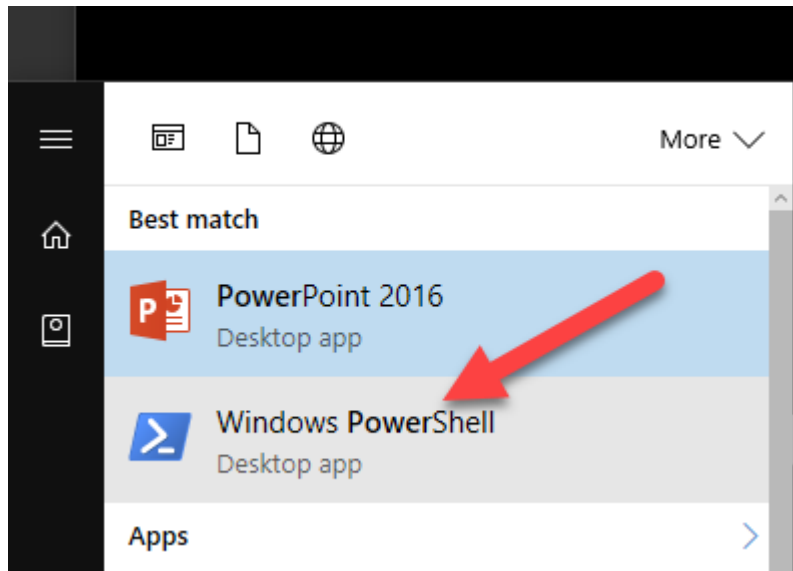
Pasos detallados



De windows golpeó la tecla de windows y nodo tipo o PowerShell



Pasos detallados



2. A continuación queremos crear una carpeta para nuestro proyecto y agregar algunos archivos iniciales (personalmente tengo todas las carpetas de proyecto en una carpeta llamada proyectos (C:\Projects en PC o mi 'userfolder' / proyectos en Mac).

Crear un directorio

```
~ $ mkdir botworkshop
```

Ir al directorio

```
~ $ cd botworkshop
```

Inicializar un archivo **package.json** . Vamos a utilizar la **-y** por lo que él nos da los valores por defecto.

```
~ $ npm init -y
```

Instalar el módulo de gestión de botbuilder.

```
~ $ npm install botbuilder --save
```

por último, queremos abrir a todos de este hasta en VSCode t y PE lo siguiente (el código de la palabra con un espacio luego de un período)

```
~ code .
```

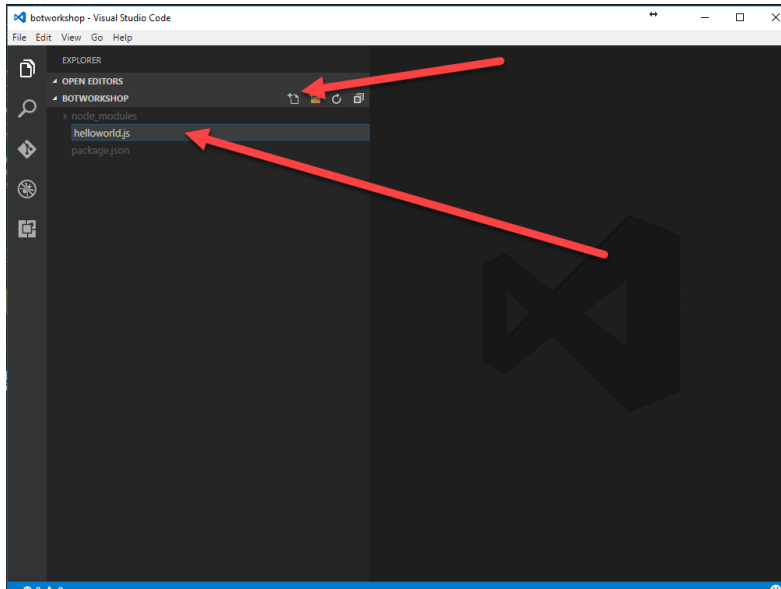
Nota : También puede abrir Visual Studio Code y seleccionar Archivo ➤ Abrir archivo (MAC) ➤ Abra la carpeta (PC) y seleccione la carpeta botworkshop que creó)

Pasos detallados

Esto abrirá nuestro proyecto en Visual Studio Code (otra vez, usted es libre de elegir su propio editor si lo desea).

Nota: Una vez en Visual Studio Code puede presionar CTRL + ' para abrir una ventana de terminal dentro del programa

3. Ahora creamos nuestro primer bot. Hacer click en el ícono para añadir archivo, helloworld.js



4. En el archivo que acaba de crear, escriba o pegue el siguiente código.

Nota : Si está utilizando el archivo PDF, puede encontrar los fragmentos de código en un archivo independiente denominado SNIPS.txt

-----SNIP1-----

```
//Talking with the user.. as simple as possible
var builder = require('botbuilder');

var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);

bot.dialog('/', function (session) {
    session.send('Hello World');
});
```

5. Volver a la línea de comandos (powershell, etc) y escriba lo siguiente: (asegúrese de que todavía está en la carpeta de proyectos botworkshop)

```
~ $ node helloworld.js
```

Pasos detallados

Verifique no haber tenidos errores.

Escriba la palabra Hola (o cualquier cosa realmente) y debe recuperar un "Hola mundo" de tu bot.

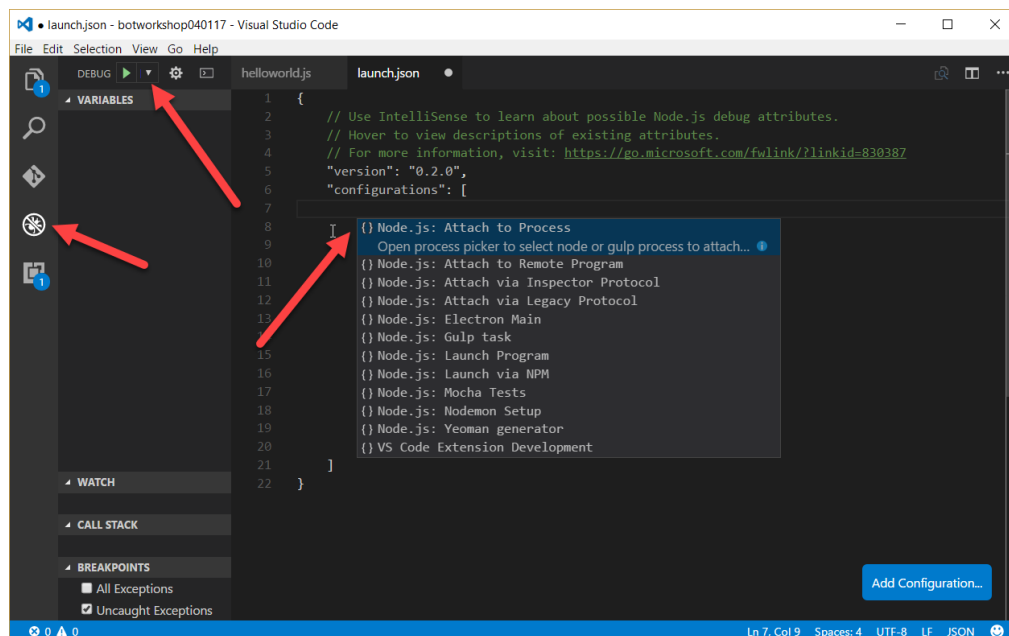
```
Windows PowerShell (Admin)

C:\projects\botworkshop
λ node helloworld.js
Hello
Hello World
C:\projects\botworkshop
λ
```

6. Interrumpa con **Ctrl+C** para salir del bot y volver a la línea de comandos.

7. Ahora configuramos una sesión de debugging que, aunque simple, nos permita acostumbrarnos al debugger de Visual Studio Code.

- > En VSCode, haga clic en el ícono de error (lado izquierdo, segundo desde la parte inferior)
- > En la parte superior se puede ver que no hay ningún archivo de configuración.
- > Haga click en la flecha hacia abajo a la derecha del botón de play verde y seleccione "Añadir configuración"
- > Esto producirá una caída hacia abajo, seleccione Node.js



8. Esto creará un archivo **launch.json**. Abra este archivo y cambie el atributo **"program"** de **index.js** a **helloworld.js**

Pasos detallados

```
1 {
2   "version": "0.2.0",
3   "configurations": [
4     {
5       "name": "Launch",
6       "type": "node",
7       "request": "launch",
8       "program": "${workspaceRoot}/index.js",
9       "stopOnEntry": false,
10      "args": [],
11      "cwd": "${workspaceRoot}",
12      "preLaunchTask": null,
13      "runtimeExecutable": null,
14      "runtimeArgs": [
15        "--no-lazy"
16      ],
17      "env": {
18        "NODE_ENV": "development"
19      },
20      "externalConsole": false,
21      "sourceMaps": false,
22      "outDir": null
23    },
24    {
25      "name": "Attach",
26      "type": "node",
27      "request": "attach",
28      "port": 5858,
29      "address": "localhost",
30      "restart": false,
31      "sourceMaps": false,
32      "outDir": null,
33      "localRoot": "${workspaceRoot}",
34    }
35  ]
36 }
```

Nota : El último paso no se utiliza en este HOL pero es bueno estar al tanto de ese campo para el futuro.

9. A continuación abra el archivo **helloworld.js** y ponga un breakpoint al lado de la línea de **session.send ('Hello World')**; haciendo clic al lado del número de línea.

```
1 var builder = require('botbuilder');
2
3 var connector = new builder.ConsoleConnector().listen();
4 var bot = new builder.UniversalBot(connector);
5
6 bot.dialog('/', function (session) {
7   session.send('Hello world');
8 })
```

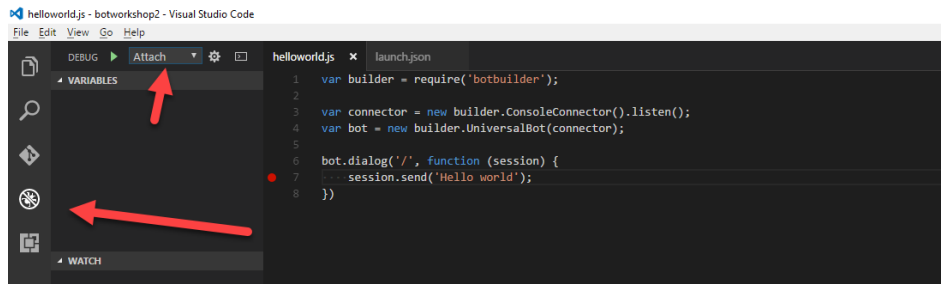
10. Ejecutamos nuestro programa en modo de depuración. Abra la consola (Terminal, Powershell, etcetera) y escriba lo siguiente:

```
~ node --debug-brk helloworld.js
```

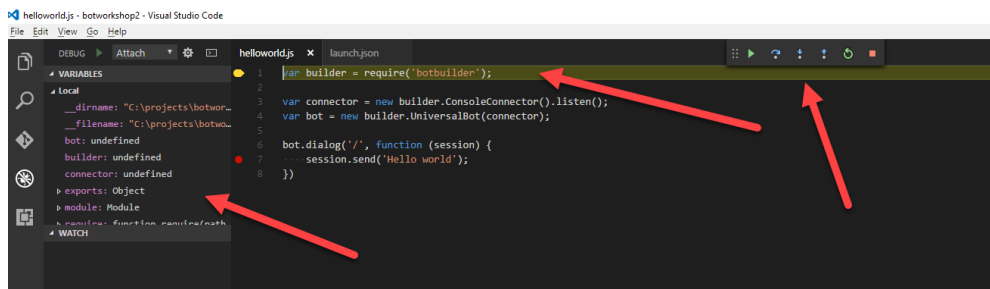
Por defecto, el proceso escuchará en el puerto 5858.

11. En Visual Studio Code, asegúrese de que usted todavía está en el panel de depuración, en el menú desplegable de depuración. Asegúrese de que selecciona **adjuntar** (para colocar en el proceso que acaba de comenzar que se está ejecutando en Puerto 5858) y haga clic en la flecha verde debug para ejecutarlo.

Pasos detallados



12. Esto adjuntará al proceso y la depuración se detendrá en la primera línea del programa (que es lo que uso **--debug-brk** hace como solo **--debug**) usted podrá ver todas las variables locales a la izquierda, navegar el código (teclas o botones en la parte superior) e inspeccionar las variables posándose sobre ellas. Puedes pulsar F5 para ejecutar el programa (o la flecha verde en la barra superior).



Pasar tiempo de depuración y mirando a su alrededor en este sencillo ejemplo por lo que se pueden depurar más complejas más adelante.

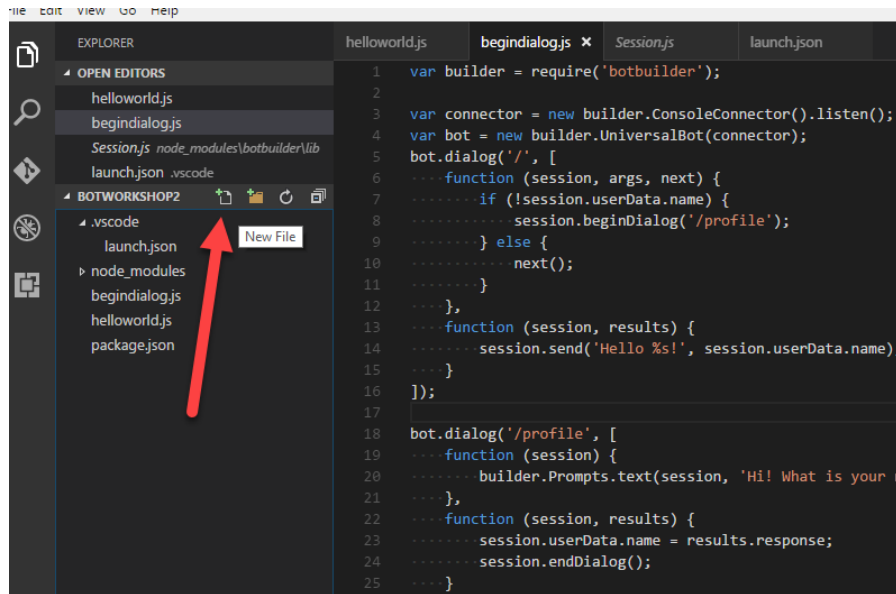
Es el fin del primer ejercicio. Ahora que ya tenemos todo configurado y un simple bot funcionando vamos a hablar sobre como manejar uno más complejo.

Ejercicio 2: Utilizando mensajes en un bot

En este ejercicio crearemos un bot muy simple que podrá ejercitar utilizando diferentes mensajes en cascada y recogida de datos.

Pasos detallados

1. Abrir el código de Visual Studio en la misma carpeta (proyecto) del ejercicio anterior. Agregar un archivo nuevo haciendo clic en el icono de archivo nuevo y asígnele el nombre **promptbot.js**



2. En primer lugar, tenemos que instanciar el objeto bot, como lo hicimos antes. Agregue el código siguiente al principio del archivo **promptbot.js**.

-----SNIP2-----

```
var builder = require('botbuilder');  
  
var connector = new builder.ConsoleConnector().listen();  
var bot = new builder.UniversalBot(connector);
```

3. A continuación tenemos que añadir nuestro diálogo. Es un diálogo, así que analicemos todo esto a la vez. (Facilita el copiar/pegar también) Pegue el siguiente código justo debajo de nuestra variable bot (**bot var = new...**) en **promptbot.js**

-----SNIP3-----

```
bot.dialog('/', [  
  function (session) {  
    builder.Prompts.text(session, "Hello... What's your name?");  
  },  
  function (session, results) {  
    session.userData.name = results.response;  
    builder.Prompts.number(session, "Hi " + results.response + ",  
How many years have you been coding?");  
  },  
  function (session, results) {
```

Pasos detallados

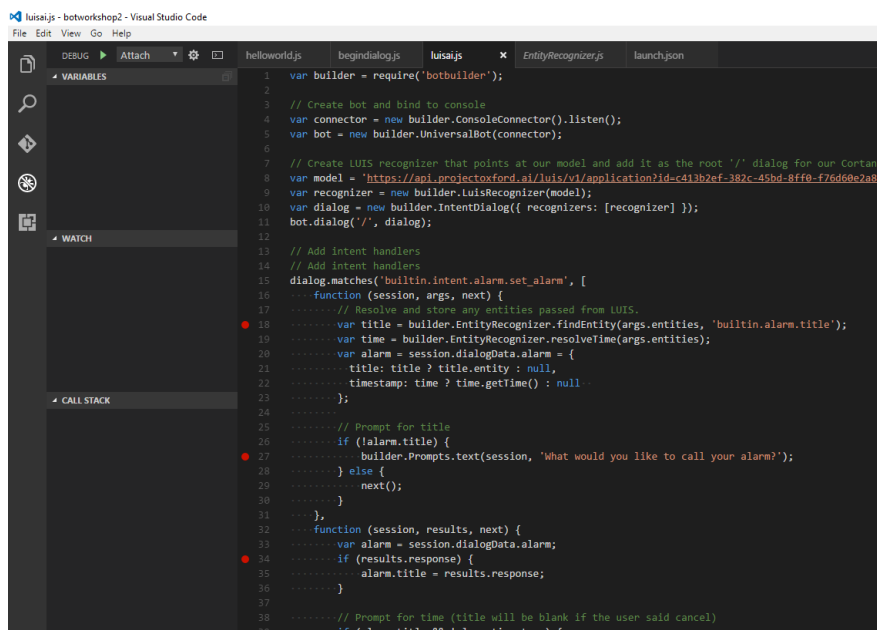
```
    session.userData.coding = results.response;
    builder.Prompts.choice(session, "What language do you code Node
using?", ["JavaScript", "CoffeeScript", "TypeScript"]);
  },
  function (session, results) {
    session.userData.language = results.response.entity;
    session.send("Got it... " + session.userData.name +
      " you've been programming for " +
session.userData.coding +
      " years and use " + session.userData.language +
".");
  }
});
```

En este código estamos creando un function array (entre los []) para que “cascadee” de una pregunta a la siguiente. Dentro de la cascada estamos utilizando 3 diferentes mensajes (texto, número, opción) y almacenando los datos del usuario. Nos sirven los datos para el mensaje final **session.send**.

4. Ahora podemos ejecutar el código. Vaya a su símbolo y escriba en el siguiente.

```
~ node --debug promptbot.js
```

5. A continuación ir a código de Visual Studio y colocar algunos puntos de interrupción en el código para que puedas paso a paso y examinar los objetos y el flujo.



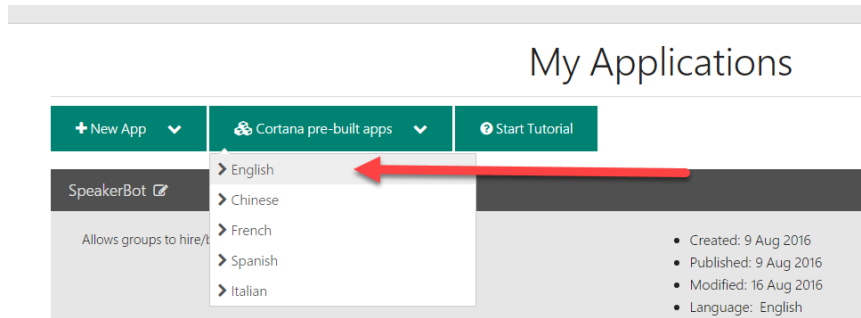
Pasos detallados
6. Paso a través del código, ya que teníamos antes ver como están funcionando las cosas.

Ejercicio 3: Uso de diálogos de LUIS

En este ejercicio crearemos un modelo LUIS. Como comentamos en la charla, podemos crear nuestro propio dominio modelos específicos, pero para este laboratorio usamos Cortana pre-compilados en la aplicación. La ventaja es que podemos ver rápidamente el poder del lenguaje natural en nuestro bot sin el entrenamiento. Esto creará un asistente personal. Para ver todo lo que este modelo prediseñado puede ver la documentación aquí: <https://www.luis.ai/Help/Index#PreBuiltApp>

Pasos detallados

1. Regístrate en <http://www.LUIS.ai>. Debe seleccionar esto para arriba en el primer ejercicio, si no ir a la sección primera.
2. Desde tu panel de control Seleccione las **Cortana pre-built apps** → Inglés



3. Una vez que hace click en su idioma, aparecerá un formulario del diálogo modelo. **Copiar la URL**, que la necesitaremos para nuestra aplicación.

×

Selected Cortana Models Prebuilt App

Help

The pre-built personal assistant V2 can interpret commands like “set an alarm for 6 pm” or “call mom”. For a full list of functionality, see the [documentation page](#) for this pre-built application.

Query

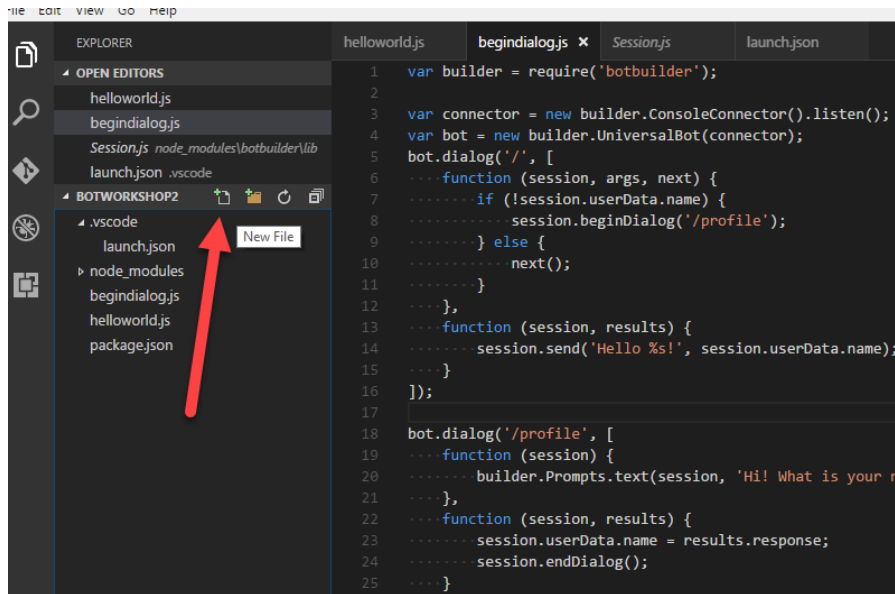
URL

[https://api.projectoxford.ai/luis/v1/application?id=c413b2ef-\[redacted\]-f76d60e2a821&subscription-key=fe054e042fd14754-\[redacted\]a5&q=](https://api.projectoxford.ai/luis/v1/application?id=c413b2ef-[redacted]-f76d60e2a821&subscription-key=fe054e042fd14754-[redacted]a5&q=)

Advanced

▼

4. Abrir el código de Visual Studio y en su proyecto; crear un archivo luisai.js



5. Agregue el código siguiente al archivo **luisai.js** .

-----SNIP4-----

```
var builder = require('botbuilder');

// Create bot and bind to console
var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);

// Create LUIS recognizer that points at our model and add it as the
root '/' dialog for our Cortana Bot.
var model = '<your models url>';
var recognizer = new builder.LuisRecognizer(model);
var dialog = new builder.IntentDialog({ recognizers: [recognizer] });
bot.dialog('/', dialog);

// Add intent handlers
dialog.matches('builtin.intent.alarm.set_alarm',
builder.DialogAction.send('Creating Alarm'));
dialog.matches('builtin.intent.alarm.delete_alarm',
builder.DialogAction.send('Deleting Alarm'));
dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't
understand. I can only create & delete alarms."));
```

6. Reemplazar '**<your models url>**' con la **URL** que copiamos en el paso 3
7. Pasar a la consola y ejecutar el siguiente

```
~ $ node luisai.js
```

8. Usted puede pedir para crear o eliminar alarmas para usted. Probar diferentes maneras de decir y aviso que se puede pedir lo mismo de muchas maneras diferentes y entiende (PNL).

```
C:\projects\botworkshop>
λ node luisai.js
help
I'm sorry I didn't understand. I can only create & delete alarms.
create an alarm for 5 minutes from now
Creating Alarm
what alarms do I have set
I'm sorry I didn't understand. I can only create & delete alarms.
do an alarm for 10 oclock
Creating Alarm
11pm alarm please
```

Ahora solo estamos recogiendo intenciones, no estamos realmente generando ninguna acción excepto para imprimir en las cosas de la pantalla como **'Creación y alarma'** o **'Borrar una alarma'**. Vamos a añadir alguna funcionalidad. Esto nos permitirá inspeccionar los intentos de LUIS y responder a ellos. Vamos a empezar con el. **set_alarm intención**

Así que reemplace el siguiente código:

Replace this whole section

```
13 // Add intent handlers
14 dialog.matches('builtin.intent.alarm.set_alarm', builder.DialogAction.send('Creating Alarm'));
15 dialog.matches('builtin.intent.alarm.delete_alarm', builder.DialogAction.send('Deleting Alarm'));
16 dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't understand. I can only create & delete alarms.));
17
```

Con este código:

-----SNIP5-----

```
// Add intent handlers
dialog.matches('builtin.intent.alarm.set_alarm', [
  function (session, args, next) {
    // Resolve and store any entities passed from LUIS.
    var title = builder.EntityRecognizer.findEntity(args.entities, 'builtin.alarm.title');
    var time = builder.EntityRecognizer.resolveTime(args.entities);
    var alarm = session.dialogData.alarm = {
      title: title ? title.entity : null,
      timestamp: time ? time.getTime() : null
    };

    // Prompt for title
    if (!alarm.title) {
      builder.Prompts.text(session, 'What would you like to call your alarm?');
    } else {
```

```

        next();
    }
},
function (session, results, next) {
    var alarm = session.dialogData.alarm;
    if (results.response) {
        alarm.title = results.response;
    }

    // Prompt for time (title will be blank if the user said cancel)
    if (alarm.title && !alarm.timestamp) {
        builder.Prompts.time(session, 'What time would you like to set the alarm
for?');
    } else {
        next();
    }
},
function (session, results) {
    var alarm = session.dialogData.alarm;
    if (results.response) {
        var time = builder.EntityRecognizer.resolveTime([results.response]);
        alarm.timestamp = time ? time.getTime() : null;
    }

    // Set the alarm (if title or timestamp is blank the user said cancel)
    if (alarm.title && alarm.timestamp) {
        // Save address of who to notify and write to scheduler.
        alarm.address = session.message.address;
        alarms[alarm.title] = alarm;

        // Send confirmation to user
        var date = new Date(alarm.timestamp);
        var isAM = date.getHours() < 12;
        session.send('Creating alarm named "%s" for %d/%d/%d %d:%02d%s',
            alarm.title,
            date.getMonth() + 1, date.getDate(), date.getFullYear(),
            isAM ? date.getHours() : date.getHours() - 12, date.getMinutes(), isAM ?
'am' : 'pm');
    } else {
        session.send('Ok... no problem.');
```

```
});
```

En la sección anterior del código, estamos utilizando una serie de técnicas que hemos analizado, como **intención de emparejar, mensajes de texto, indicaciones de tiempo y una cascada**. A continuación tenemos que añadir un intención de diálogo para eliminar una alarma.

Pegue el siguiente código **debajo de** la última sección en **luisai.js**.

-----SNIP6-----

```
dialog.matches('builtin.intent.alarm.delete_alarm', [
  function (session, args, next) {
    // Resolve entities passed from LUIS.
    var title;
    var entity = builder.EntityRecognizer.findEntity(args.entities,
    'builtin.alarm.title');
    if (entity) {
      // Verify its in our set of alarms.
      title = builder.EntityRecognizer.findBestMatch(alarms,
    entity.entity);
    }

    // Prompt for alarm name
    if (!title) {
      builder.Prompts.choice(session, 'Which alarm would you like
    to delete?', alarms);
    } else {
      next({ response: title });
    }
  },
  function (session, results) {
    // If response is null the user canceled the task
    if (results.response) {
      delete alarms[results.response.entity];
      session.send("Deleted the '%s' alarm.",
    results.response.entity);
    } else {
      session.send('Ok... no problem.');
```

Como se puede ver, es muy similar a la sección de alarma add con la adición de utilizar el **indicador de elección**. Ahora tenemos que añadir dos fragmentos más para hacerla completa. Necesitamos agregar nuevamente nuestra sección default (una línea de código) y una aplicación muy simple de una alarma. Pegue el código siguiente en la **parte inferior** del archivo **luisai.js**.

-----SNIP7-----

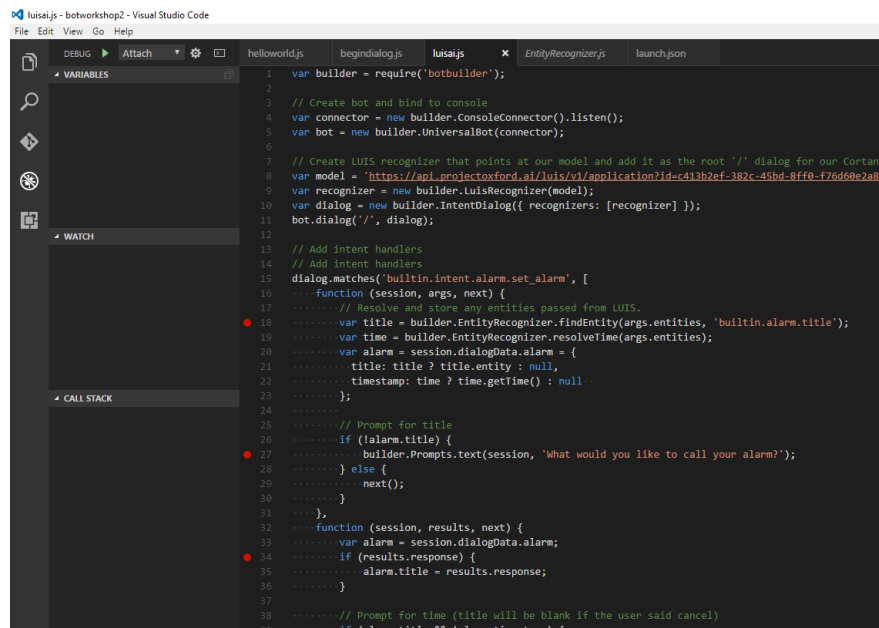
```
dialog.onDefault(builder.DialogAction.send("I'm sorry I didn't
understand. I can only create & delete alarms.));

// Very simple alarm scheduler
var alarms = {};
setInterval(function () {
    var now = new Date().getTime();
    for (var key in alarms) {
        var alarm = alarms[key];
        if (now >= alarm.timestamp) {
            var msg = new builder.Message()
                .address(alarm.address)
                .text("Here's your '%s' alarm.", alarm.title);
            bot.send(msg);
            delete alarms[key];
        }
    }
}, 15000);
```

9. Ahora podemos ejecutar el código. Vaya a su símbolo y escriba en el siguiente.

```
~ node --debug luisai.js
```

10. A continuación ir a código de Visual Studio y colocar algunos puntos de interrupción en el código para que puedas paso a paso y revise las entidades, como se están estableciendo.



11. Paso a través del código, ya que teníamos antes ver como están funcionando las cosas.

Ejercicio 4: Conexión a Skype y Webchat

En este ejercicio vamos a conectar Skype y un canal de Webchat a tu bot. De esta manera, no vamos a hostear nuestro bot en la nube sino accediéndolo desde nuestros dispositivos móviles. Estamos haciendo esto porque:

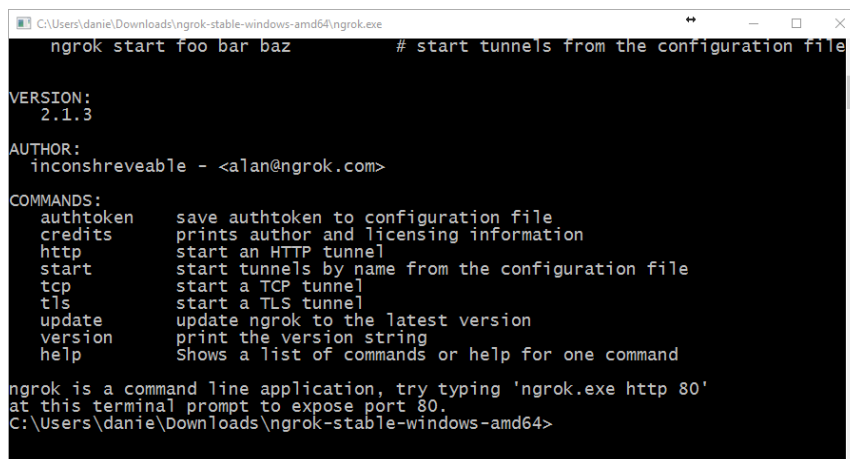
- 1) reduce la complejidad del taller por no tener que solucionar problemas de cargar y ejecutar una nodo de la aplicación en su nube de elección.
- 2) sirve para probar y diagnosticar problemas en el bot.

Nota: Una alternativa para las pruebas es utilizar el emulador de bots, que es un código abierto: <https://docs.botframework.com/en-us/tools/bot-framework-emulator/>

Lo primero que debemos hacer es ejecutar NGrok. Si descargó NGrok de <https://ngrok.com/>, NGrok le permite **crear «Túneles seguros a su localhost»**. Esto significa que podemos exponer en Internet nuestro bot y hacerlo accesible desde nuestro móvil.

Pasos detallados

1. Descargar y descomprimir Ngrok en su computadora
2. Una vez descomprimido haga doble clic en el archivo **ngrok.exe** . Esto abrirá su propia ventana de comandos.



```
C:\Users\danie\Downloads\ngrok-stable-windows-amd64\ngrok.exe
ngrok start foo bar baz # start tunnels from the configuration file

VERSION:
  2.1.3

AUTHOR:
  inconshreveable - <alan@ngrok.com>

COMMANDS:
  authtoken  save authtoken to configuration file
  credits    prints author and licensing information
  http       start an HTTP tunnel
  start      start tunnels by name from the configuration file
  tcp        start a TCP tunnel
  tls        start a TLS tunnel
  update     update ngrok to the latest version
  version    print the version string
  help       Shows a list of commands or help for one command

ngrok is a command line application, try typing 'ngrok.exe http 80'
at this terminal prompt to expose port 80.
C:\Users\danie\Downloads\ngrok-stable-windows-amd64>
```

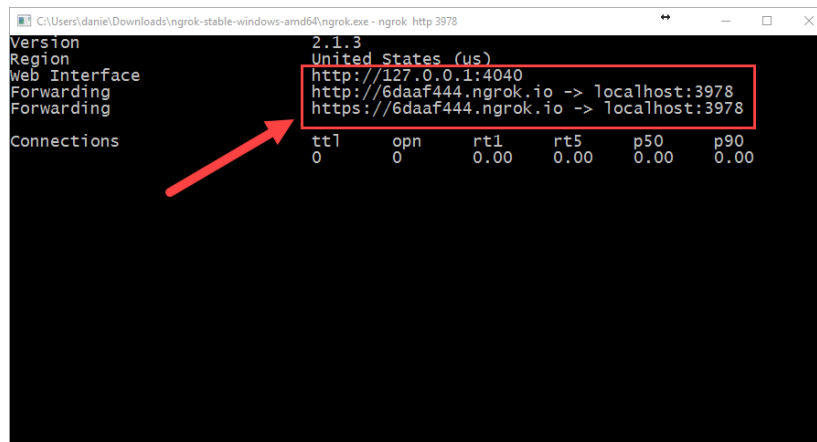
Pasos detallados

3. Ejecutar el siguiente comando:

```
~ $ ngrok http 3978
```

Nota: en un MAC, necesitará escribir `. / ngrok http 3978`

Debería ver lo siguiente en la ventana de comandos.



```
C:\Users\danie\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 3978
Version                2.1.3
Region                 United States (us)
Web Interface          http://127.0.0.1:4040
Forwarding              http://6daaf444.ngrok.io -> localhost:3978
                       https://6daaf444.ngrok.io -> localhost:3978
Connections
  ttl    opn    rt1    rt5    p50    p90
    0     0     0.00  0.00  0.00  0.00
```

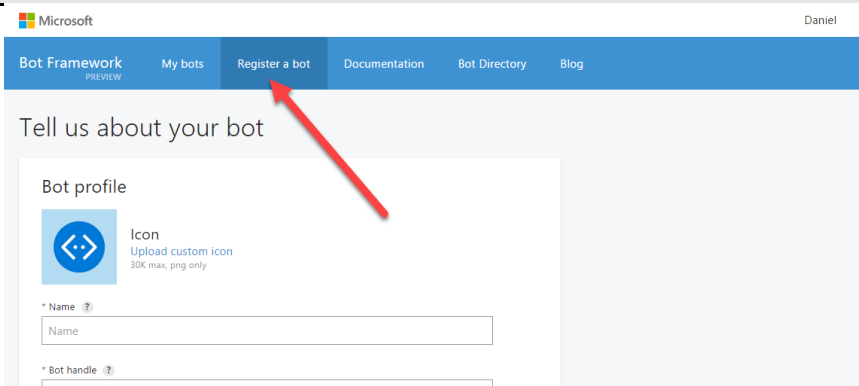
Nos preocupamos acerca de dos cosas en esta ventana.

1. el reenvío de URL (http y https) que serán nuestra dirección externa para nuestro bot llegar a nuestra máquina local.
2. la dirección de la interfaz Web. Vamos a utilizar esto para rastrear el tráfico que llega a este puerto para nuestro bot.

Deje este. Vamos a necesitar esto para el resto de este tutorial.

4. Ahora necesitamos establecer nuestro bot en la página BotFramework. Abra un navegador y vaya a <http://BotFramework.com> . Si ya configuró una cuenta al principio de este laboratorio.
5. Si no está todavía logueado, hágalo con la Microsoft Account que creó anteriormente. Cuando entre, haga click en el registro de un bot en el menú:

Pasos detallados

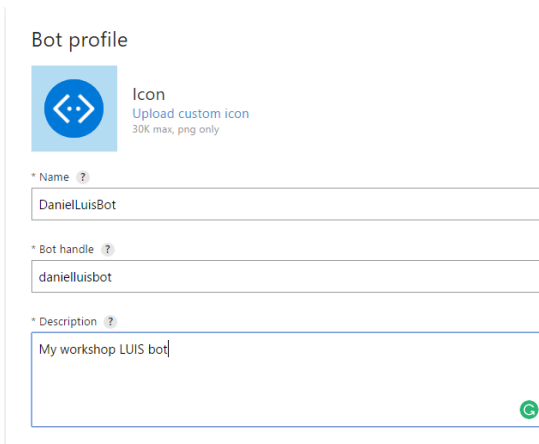


6. Rellene el formulario de registro de bot. Iremos a través de su sección por sección y tocar en los campos importantes.

Nombre: <nombre del bot>

Bot handle: <secuencia alfanumérica de caracteres> se utilizará en el SDK de C# al hacer referencia a tu bot (no en Node.js SDK)

Descripción: <alguna descripción>



7. La siguiente sección consiste en completar dos campos pero implica varios pasos. El primer cuadro es el punto final de mensajería. Si usted hospeda el bot en la nube, entonces esta sería la dirección del sitio que lo aloja. Algo como `http://DanielSpeakerBot.com/api.messages` pero ya que estamos organizando lo localmente necesitamos utilizar la dirección que Ngrok nos dio cuando lo usamos.

Pasos detallados

```
C:\Users\danie\Downloads\ngrok-stable-windows-amd64\ngrok.exe - ngrok http 3978
Version 2.1.3
Region United States (us)
Web Interface http://127.0.0.1:4040
Forwarding http://6daaf444.ngrok.io -> localhost:3978
Forwarding https://6daaf444.ngrok.io -> localhost:3978
Connections
  ttl    opn    rt1    rt5    p50    p90
    0     0     0.00   0.00   0.00   0.00
```

Añadir la dirección de Ngrok con /api/messages

Configuration

Messaging endpoint:

Register your bot with Microsoft to generate a new App ID and password

Create Microsoft App ID and password

Paste your app ID below to continue

Luego necesitas crear un AppID y la contraseña de tu bot. Haga clic en el botón "crear Microsoft App ID y contraseña".

Configuration

Messaging endpoint:

Register your bot with Microsoft to generate a new App ID and password

Create Microsoft App ID and password

Paste your app ID below to continue

Cuando haces esto una nueva página le abrirá y le dará un **App ID**.

Generate App ID and password

App name

App ID

Generate a password to continue

GUARDAR ESTE APP ID EN ALGÚN LUGAR. QUE NECESITAREMOS MÁS ADELANTE.

Pasos detallados

A continuación, haga clic en el botón **generar una contraseña para continuar**.

Generate App ID and password

App name
DanielLuisBot

App ID
2f29ce22-...

Generate a password to continue



Esto aparecerá un cuadro de diálogo modal con su contraseña.

New password generated

This is the only time when it will be displayed. Please store it securely. Paste this password into your bot configuration file.

Mzq.....

Ok

Este es el único momento en que aparecerá.

GUARDAR ESTE ID DE CONTRASEÑA EN ALGÚN LUGAR. QUE NECESITAREMOS MÁS ADELANTE

Haga clic en **ok** para continuar.

Luego haga clic en el botón **finalizar y volver a Bot marco** para continuar.

Generate App ID and password

App name
DanielLuisBot

App ID
2f29ce22-...

Password [Learn More](#)
Mzq..... **Delete**

Finish and go back to Bot Framework



8. En la sección final, no necesitamos añadir nada. (aunque en el futuro, configurar y utilizar una clave de ideas de aplicación le dará varios informes)

Pasos detallados

Admin

Owners: ?

danielegan0112@gmail.com

Instrumentation key: ?

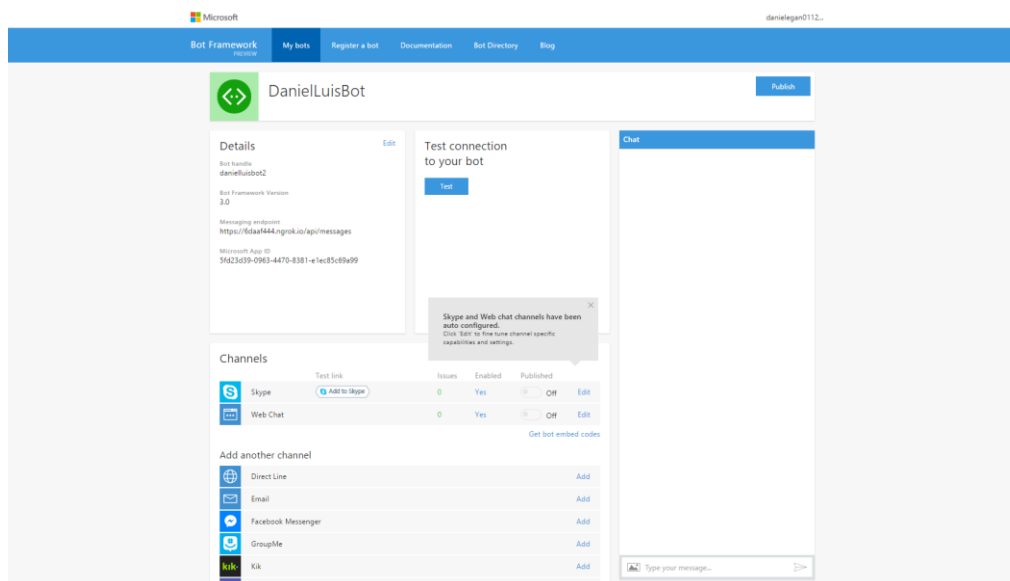
Instrumentation key (Azure App Insights key)

☐ By clicking Register, you agree to the [Privacy statement](#), [Terms of use](#), and [Code of conduct](#).

Register Cancel

9. Haga click en **registrarse** para crear el bot.

10. Deje esta página. Necesitaremos venir aquí después porque modificaremos nuestro bot para enlazarlo con el Bot Framework.



11. Abrir **luisai.js** en el Visual Studio Code. Estaremos modificando la siguiente sección de este archivo:

Pasos detallados

```
3
4  var builder = require('botbuilder');
5
6  // Create bot and bind to console
7  var connector = new builder.ConsoleConnector().listen();
8  var bot = new builder.UniversalBot(connector);
9
```

Hasta ahora, hemos estado usando el ConsoleConnector. Ahora vamos a usar el ChatConnector. Pero primero tenemos que añadir un módulo de nodo que nos ayudará con que sirve esta aplicación. Se llama Restify.

12. Abrir la consola (Terminal, Powershell, etcetera..). Asegúrese de que usted está en la carpeta botworkshop (o lo que se llama) y escriba lo siguiente.

```
~$ npm install restify --save
```

Restify es un módulo de nodo que ayuda a facilitar el resto de llamadas.

13. Una vez hecho esto, abrir el archivo luisai.js y agregue el código siguiente debajo de la constructor `var = require('botbuilder');` pegar

```
var restify = require('restify');
```

El código debería parecerse a esto

```
var builder = require('botbuilder');
var restify = require('restify');
// Create bot and bind to console
var connector = new builder.ConsoleConnector().listen();
var bot = new builder.UniversalBot(connector);
```

14. Ya que estaremos trabajando con el nuevo conector. Eliminar el conector y las líneas del bot:

Pasos detallados

```
var builder = require('botbuilder');
var restify = require('restify');

// Create bot and bind to console
var connector = new builder.ConsoleConnector({ listen() });
var bot = new builder.UniversalBot(connector);
```

Debemos configurar el servidor de restify. Para ello, agregue que las siguientes líneas debajo de la declaración de restify:

-----SNIP8-----

```
// Setup Restify Server
var server = restify.createServer();
server.listen(process.env.port || process.env.PORT || 3978, function ()
{
  console.log('%s listening to %s', server.name, server.url);
});
```

Creamos un servidor que busca una variable de entorno denominada puerto, si no encuentra uno, iniciará en Puerto **3978** (por eso usamos ese puerto para NGrok).

A continuación queremos crear el nuevo ChatConnector (en vez de ConsoleConector).

Agregue el código siguiente directamente bajo el último código pegado arriba:

-----SNIP9-----

```
// Create chat bot
var connector = new builder.ChatConnector({
  appId: process.env.MICROSOFT_APP_ID,
  appPassword: process.env.MICROSOFT_APP_PASSWORD
});
```

Aviso que requiere que el conector y **appId** y **appPassword**. Estos son lo que salvó cuando creamos nuestros bots en BotFramework.com.

Ahora necesitamos instanciar el bot y rutear apropiadamente los mensajes. Agregue este código a continuación del código anterior:

Pasos detallados

-----SNIP10-----

```
var bot = new builder.UniversalBot(connector);  
server.post('/api/messages', connector.listen());
```

Hay un último paso: añadir nuestro **appId** y **appPassword**. Para protegerlos, debe **siempre** poner en **Variables de entorno** (u otro lugar seguro). Si va a poner esto en producción o planea de guardar este código en github u otro repositorio, **NO** se debe hacer lo siguiente:

Modificar el conector para agregar su **appId** y **appPassword**.

```
// Create chat bot  
var connector = new builder.ChatConnector({  
  ... appId: '2f29ce22-...',  
  ... appPassword: 'MzqY...',  
});
```

15. Ahora permite ejecutar un bot. Ir a la consola y ejecute el siguiente comando.

```
~ $ node luisai.js
```

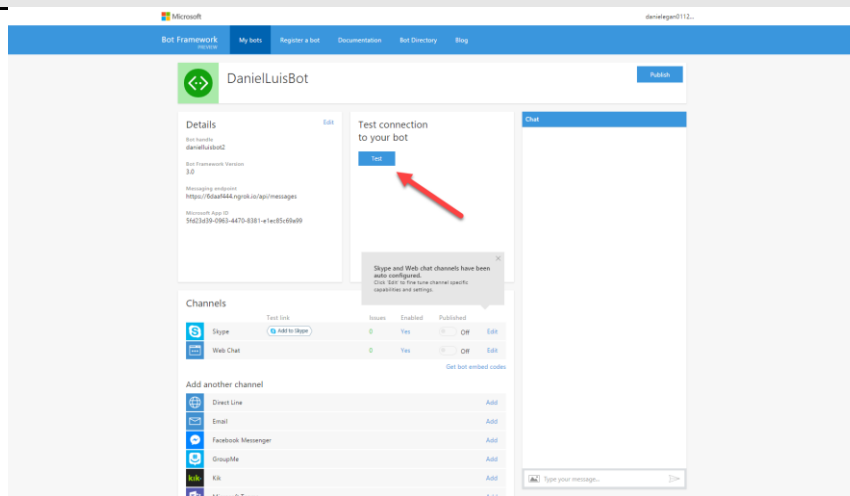
Debería ver el servidor que restify ejecuta en el puerto que especifique.

```
C:\projects\botworkshop2  
λ node luisai.js  
restify listening to http://[::]:3978  
var bot = new builder.UniversalBot(connector);  
server.post('/api/messages', connector.listen());
```

16. Podemos probar la conexión entrando a tu bot en **BotFramework.com**

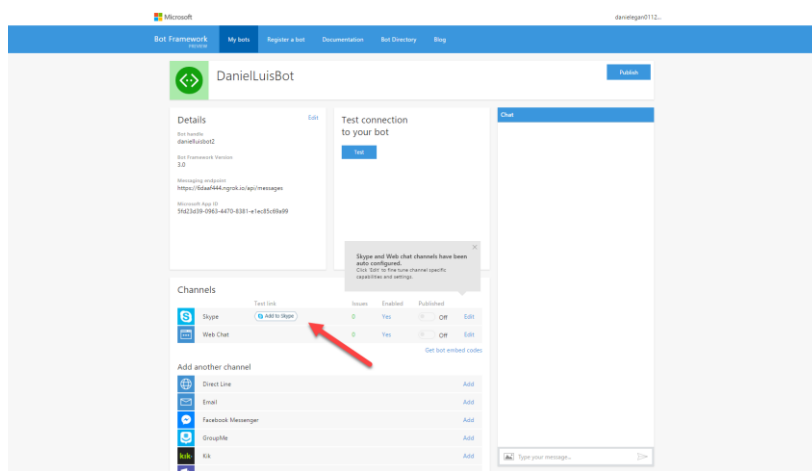
Si todo funciona bien cuando haga click en el botón de prueba, se obtendrá un mensaje de aceptación.

Pasos detallados



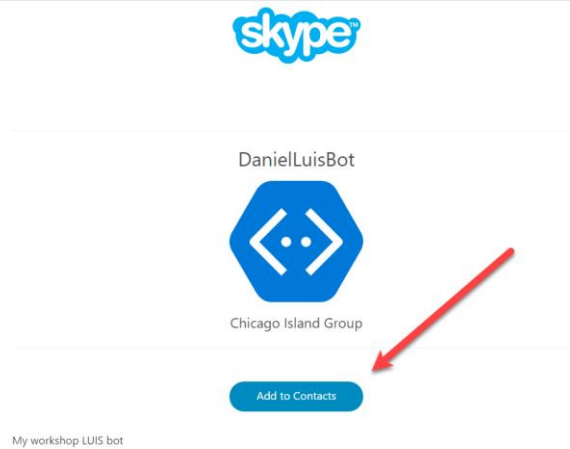
17. Ahora podemos probar Skype (debe tener Skype instalado).

En su portal, haga click en Add to Skype Button

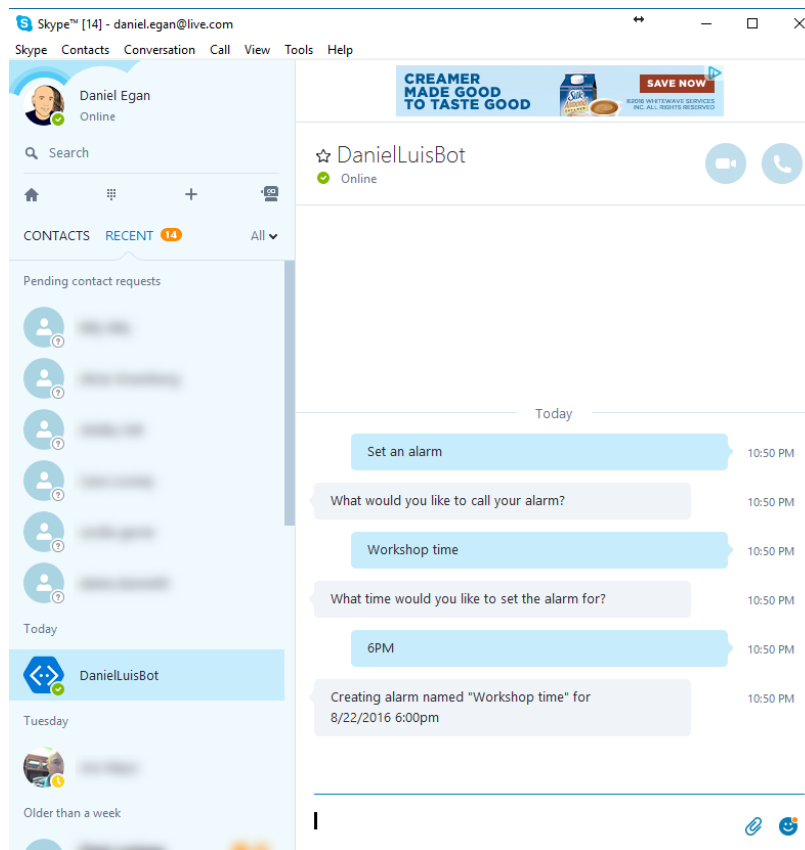


Luego, haga click en agregar usted contactos:

Pasos detallados

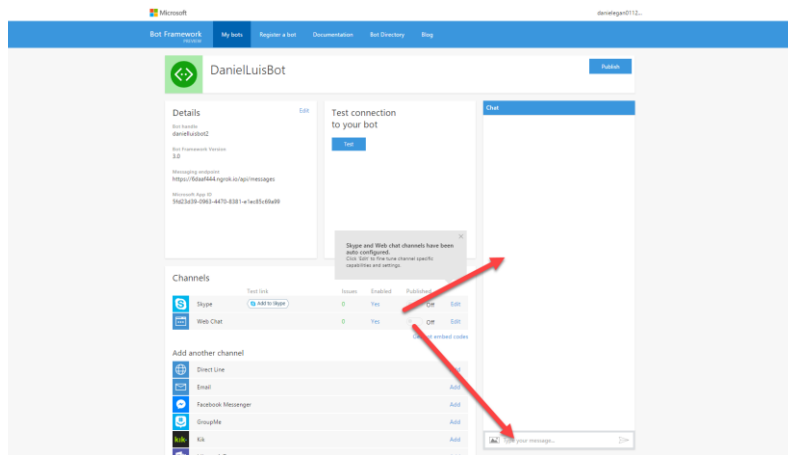


Una vez que se agrega a sus contactos, usted puede iniciar la conversación:



Si no tiene Skype instalado puede probar hacia fuera mediante una ventana de webchat. Esto está incrustado en la página de registro del bot.

Pasos detallados



Si desea que un control web local para probar con usted también puede Agregar a su proyecto. Lo primero que debemos hacer es crear una página para alojar el control webchat.

18. Abrir Visual Studio Code y añadir un archivo llamado index.html

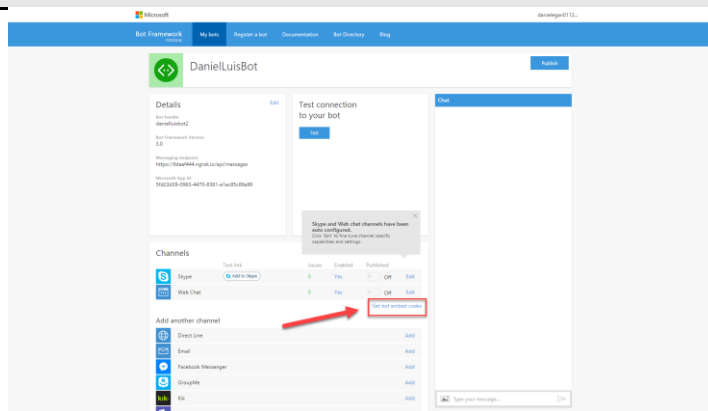
-----SNIP11-----

```
<!doctype html>
<html>
  <head>
    <title>MyAppID</title>
  </head>
  <body>

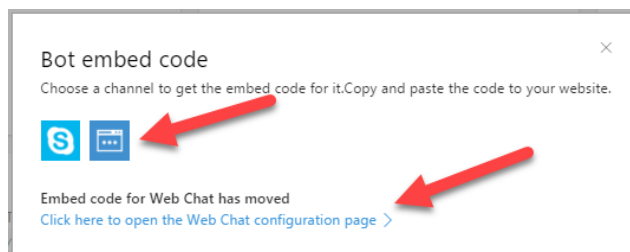
  </body>
</html>
```

19. A continuación, vaya al bot en BotFramework.com y haga clic en el bot para conseguir el código para embeber en el html, y así poder chatear con el bot desde ahí:

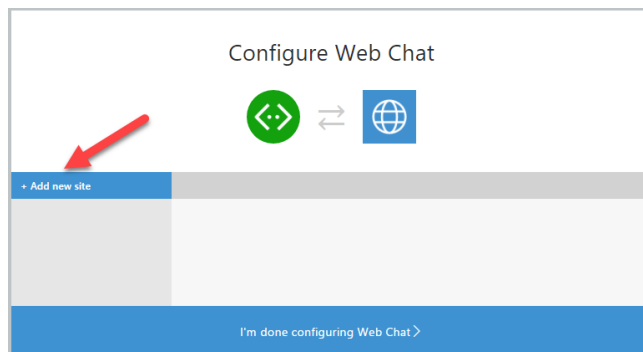
Pasos detallados



Haga click en el ícono de Chat de la Web y luego seguir el enlace.

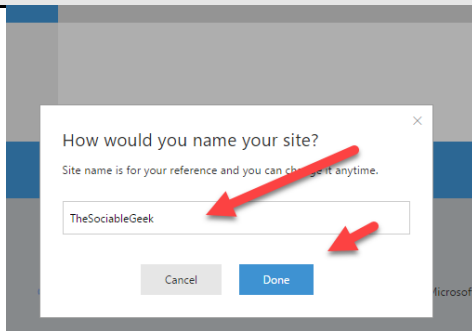


Haga clic en Agregar nuevo vínculo a sitios

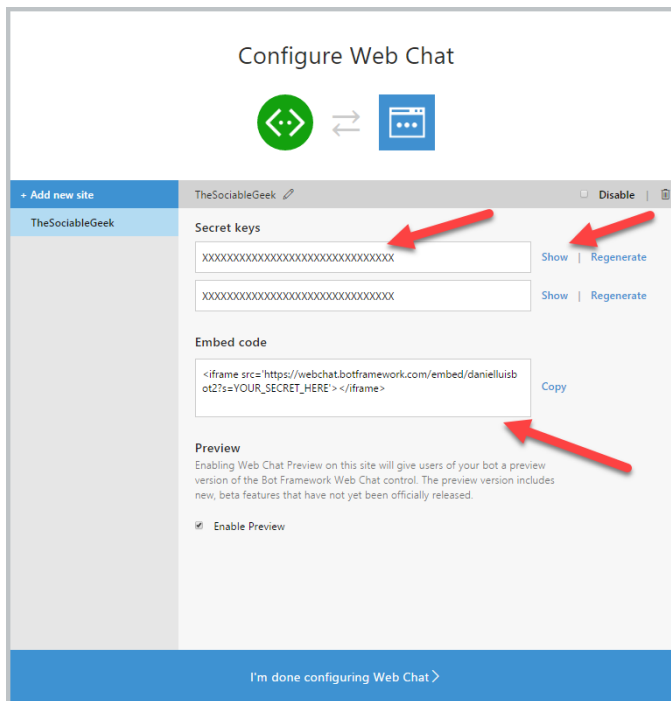


Escriba el nombre del sitio que está sucediendo. Esto es sólo para sus propósitos por lo que se puede personalizar por sitio.

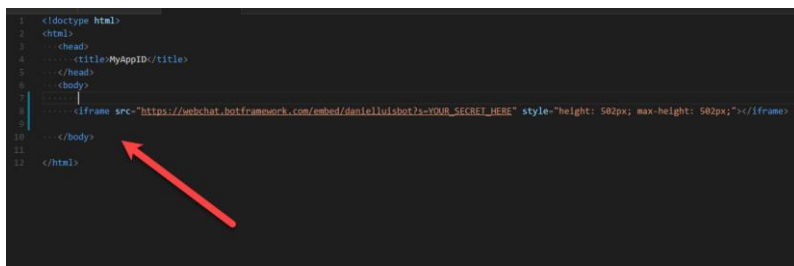
Pasos detallados



Esto genera el embedded code (IFrame) y sus claves secretas. Haga click en Mostrar en uno de ellos para que puedas copiarlo y guardarlo para su uso en el siguiente paso.




Copiar el embedded code en la sección del cuerpo en el index.html que creó.



Vuelva a colocar la sección YOUR_SECRET_KEY con la clave que ha guardado.

Pasos detallados

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>MyAppID</title>
5   </head>
6   <body>
7
8     <iframe src="https://webchat.botframework.com/embed/daniel-luisbot?s=LdX48r9nVnVJL" style="width: 100%; height: 300px; border: 1px solid #ccc;" ></iframe>
9
10  </body>
11 </html>
```



Por último, para poder acceder a esta página, necesitará añadir un **server.get** al archivo **luisai.js** .

Debajo del **server.post ('/ api/mensajes, connector.listen());** línea, agregue el código siguiente.

-----SNIP12-----

```
Server.get ('/', restify.serveStatic({
  directory: __dirname,
  default: ' / index.html'
}));
```

Esto dirigirá las peticiones get entrantes (en el navegador) a nuestra página **index.html** .

Para probarlo, vaya a la consola y ejecute el siguiente comando:

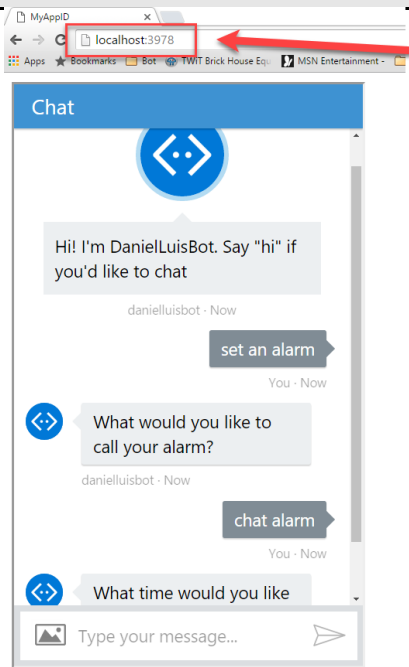
```
~ $ node luisai.js
```

Debería ver el servidor que restify ejecuta en el puerto que especifique.

```
C:\projects\botworkshop2> node luisai.js
restify listening to http://[::]:3978
```

Navegar a <http://localhost:3978>

Pasos detallados



Ahora puedes chatear.

Recursos adicionales

Derechos de autor

Información en este documento, incluyendo la dirección URL y otras referencias del sitio Web de Internet, está sujeta a cambios sin previo aviso y se proporciona sólo para fines informativos. El riesgo del uso o resultados del uso de este documento permanece con el usuario y Microsoft Corporation no otorga ninguna garantía, expresa o implícita. A menos que se indique lo contrario, las empresas, organizaciones, productos, nombres de dominio, direcciones de correo electrónico, logotipos, personas, lugares y acontecimientos citados en los ejemplos son ficticios. Ninguna asociación con compañía real, organización, producto, nombre de dominio, dirección de correo electrónico, logotipo, persona, lugar o evento se debe asociarse o inferirse. Cumplimiento de las leyes de derechos de autor es

responsabilidad del usuario. Sin limitar los derechos bajo copyright, ninguna parte de este documento puede ser reproducida, almacenada en introducido en un sistema de recuperación o transmitida en cualquier forma o por cualquier medio (electrónico, mecánico, fotocopia, grabación u otro tipo) o para cualquier propósito, sin el permiso escrito expreso de Microsoft Corporation.

Microsoft puede tener patentes, solicitudes de patentes, marcas registradas, derechos de autor u otros derechos de propiedad intelectual cubriendo el tema en este documento. Salvo lo dispuesto expresamente en cualquier acuerdo de licencia escrito de Microsoft, la decoración de este documento no le da ninguna licencia a estas patentes, marcas registradas, copyrights u otra propiedad intelectual.

© 2009 Microsoft Corporation. Todos los derechos reservados.

Microsoft y Windows son marcas comerciales del grupo Microsoft de empresas.

Todas las otras marcas registradas son propiedad de sus respectivos propietarios.