# **Alexa. Amazon Echo**

## Light Ring

### Iteración (Azul cian + blanco)

* Si no hay **ningún color activo**, está **esperando** a que hables y le des una orden/información.
* Si nombramos la “Wake Word” (Alexa/Amazon) se pondrá en **azul** (con un tono cian apuntando en la dirección del hablante).
* **Azul cian** rotatorio -> después de haber hablado, este color significa que Alexa está **procesando** **la orden**.
* **Blanco** -> cuando giremos manualmente el anillo para ajustar el volumen (**indicador de volumen**).

### Estado (rojo, morado y naranja)

* **Morado** -> **problema** durante la configuración del **wifi**.
* **Naranja** (rotatorio) -> el dispositivo se está **conectando** a su **red**.
* **Rojo** fijo -> el **micrófono** se ha **apagado** y Alexa no está escuchando los comandos.
* Luz giratoria **azul** con **destello final morado** fijo -> se ha activado el modo **“No Molestar”.**
* Destello **morado** después de interactuar con Alexa -> indica que se ha activado el modo **“No Molestar”.**

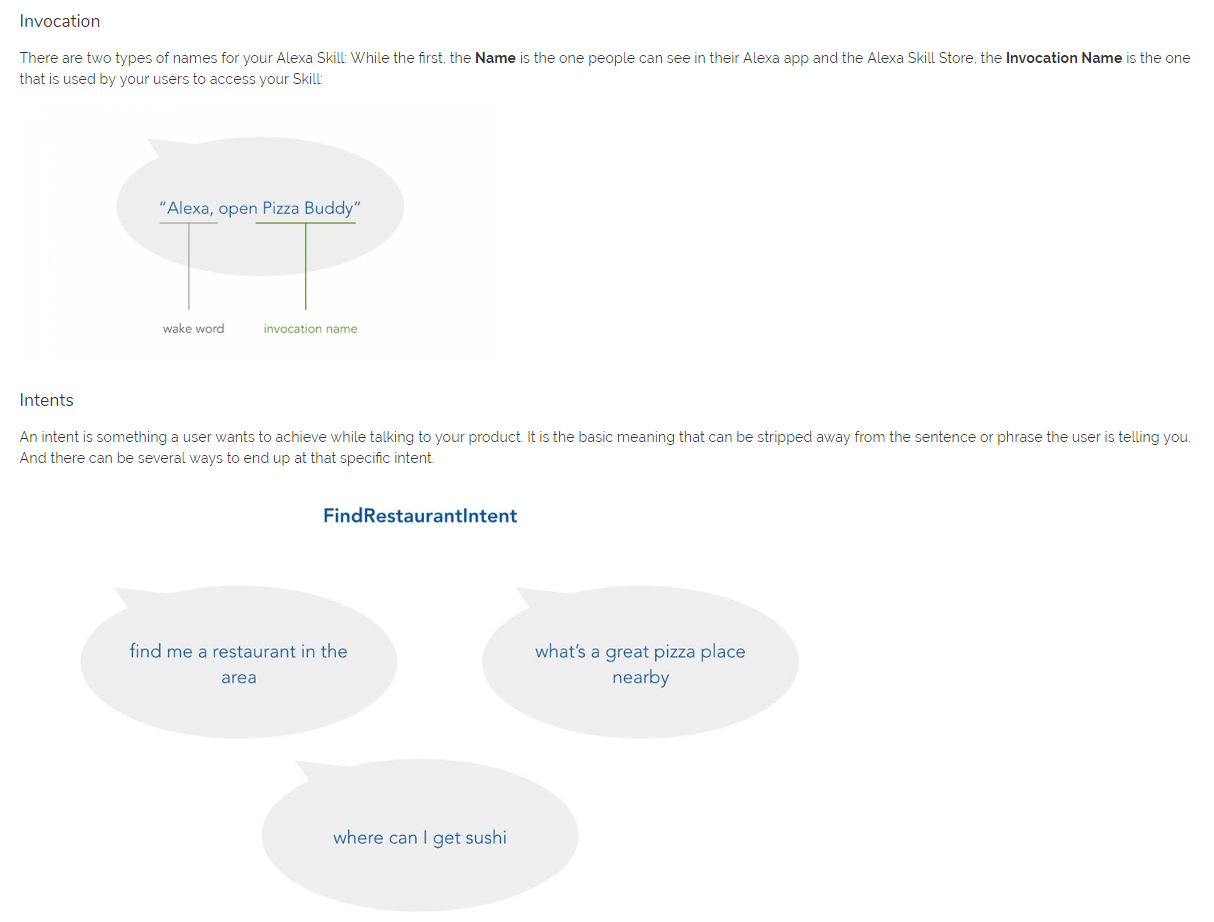
### Notificaciones (verde + amarillo)

* Luz **verde** intermitente -> **llamada** entrante.
* Luz **amarilla** intermitente -> **mensajes** en la bandeja de entrada.

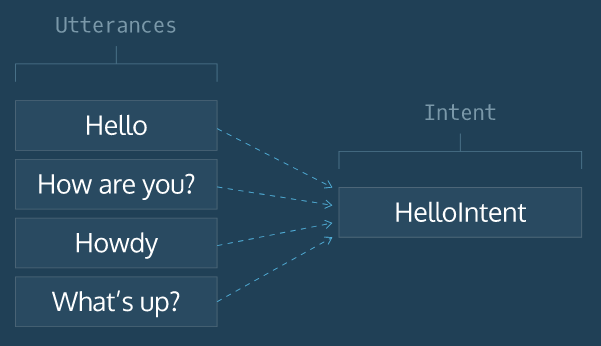
<https://www.amazon.com/gp/help/customer/display.html?nodeId=201601790>

## Alexa. Intents y utterances.

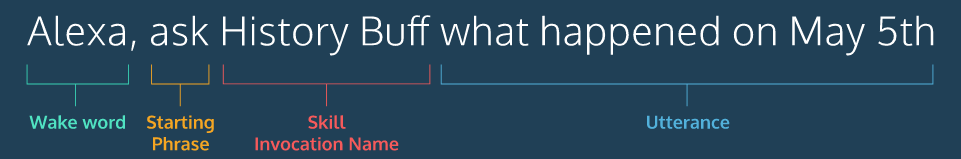
* **Intent** -> características o comportamientos que tendrá nuestra skill (invocation name). Una skill típica tendrá múltiples intents. Cada intent describe un comportamiento específico, como botones de una página web. (Nombre con el que se activará nuestra función -> **skill** (ejemplo: open **CodeAcademy**). Funciones de la skill -> **intents** (ejemplo: HelloIntent (“hello CodeAcademy”), QuestionIntent (“Do you know what is our most important language?”…))
* **Utterances** -> inputs del usuario que activarán el intent.



Ejemplo:



Ejemplo:

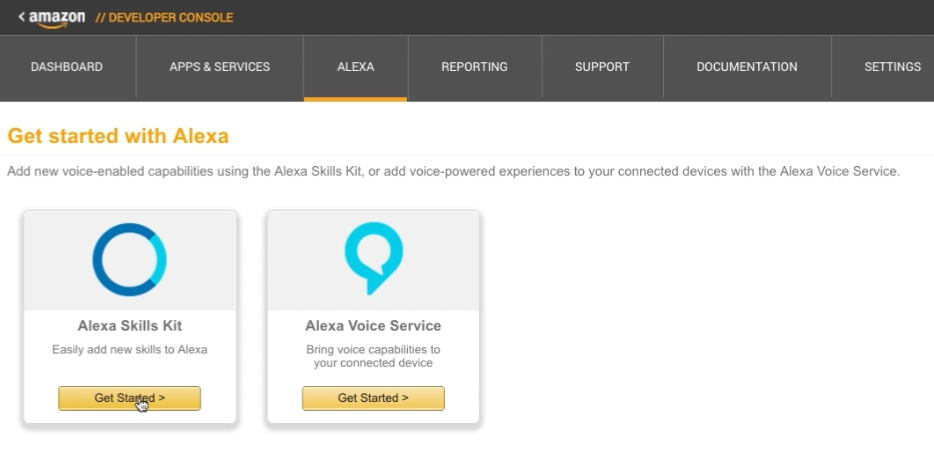


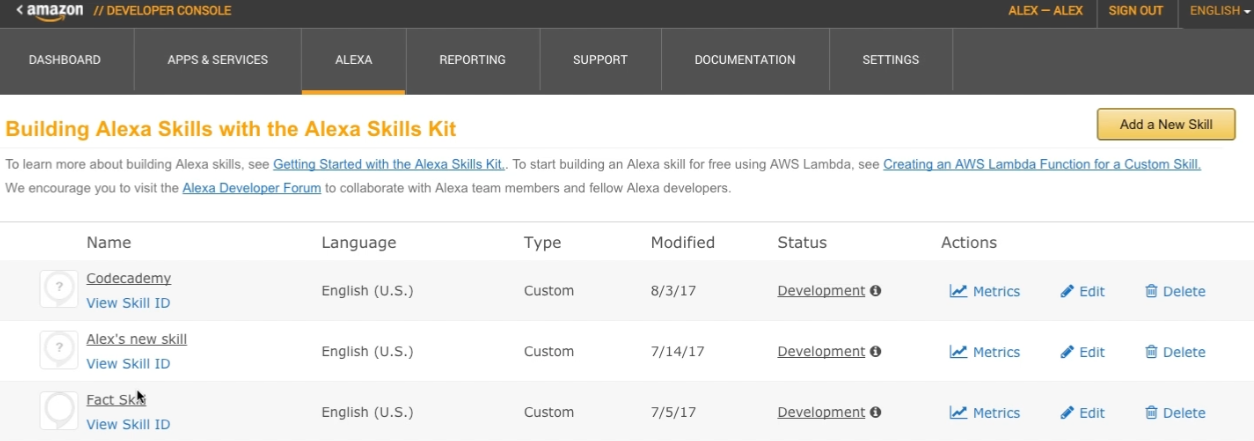
Ejemplo:

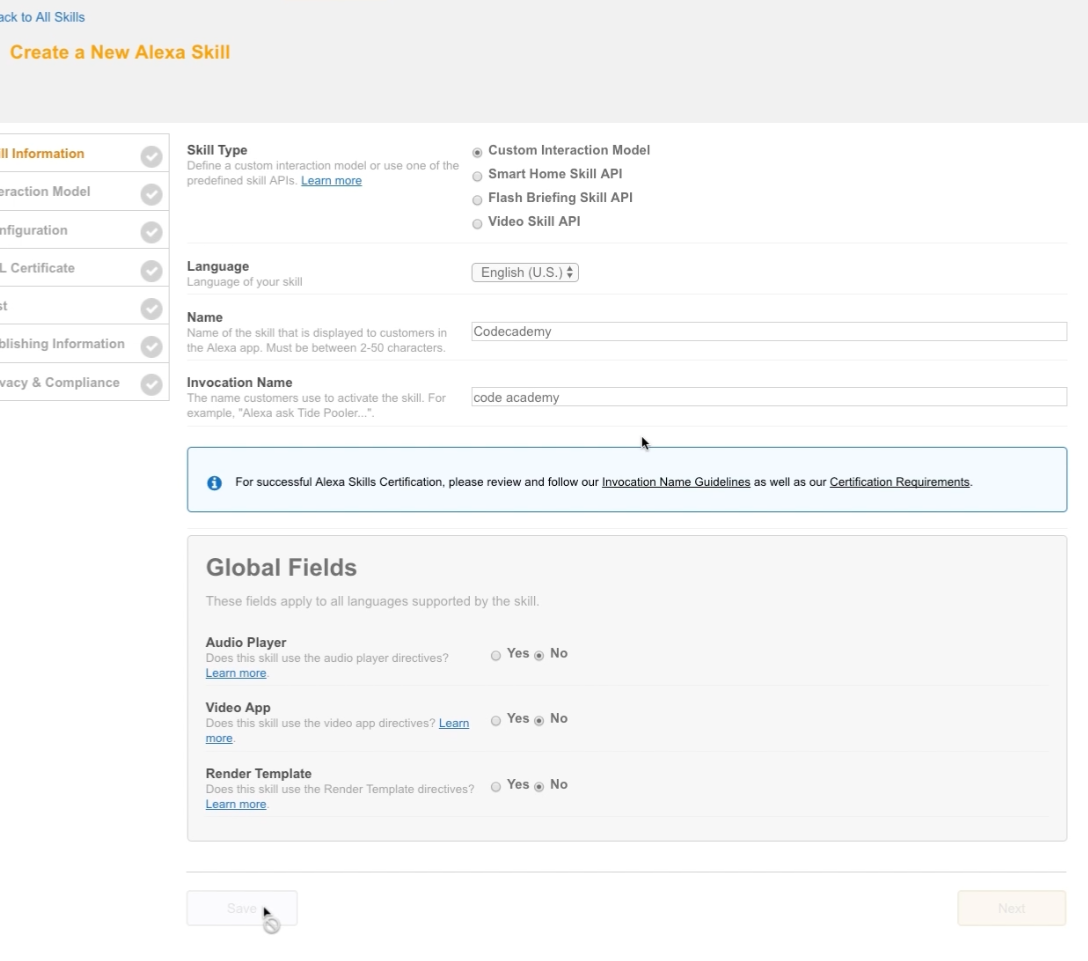
|  |  |  |  |
| --- | --- | --- | --- |
| **<STARTING PHRASE>** | **<INVOCATION NAME>** | **<PREPOSITION>** | **<SOME ACTION>** |
| Ask | Daily Horoscopes | To | Give me the horoscope |
| Tell | Google | About | For Taurus |
| Search | My sky reporter | For | The weather |
| Open | Recipies | … | How do I make an omelet |
| Load | … |  | … |
| … |  |  |  |

## Amazon Developer

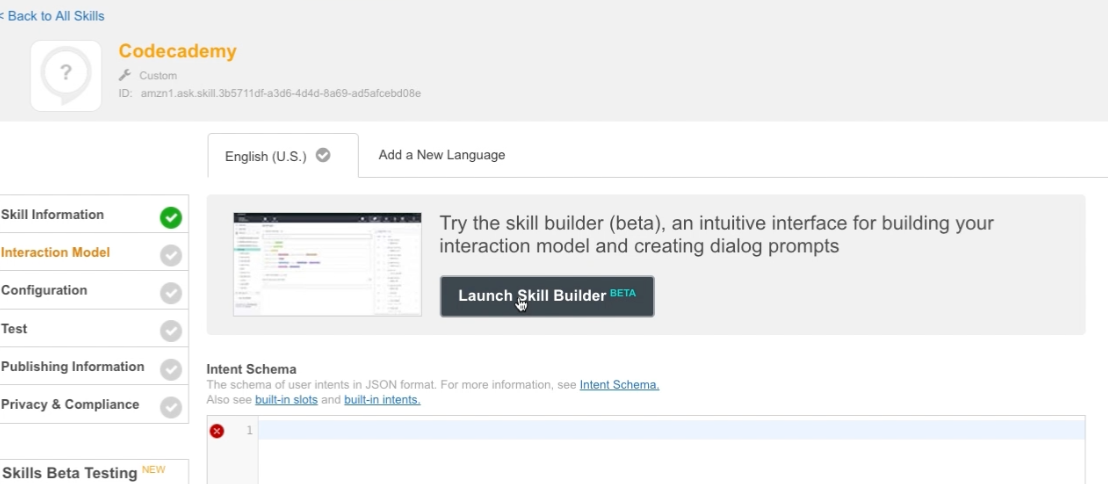
https://developer.amazon.com/alexa/console/ask/build/custom/amzn1.ask.skill.724f5e71-c4b7-4b2c-a865-b50862b78ffd/development/en\_US/dashboard

* **Front-End** -> Alexa Skills Kit/Skill Builder Beta
  + ****Nos registramos en Amazon Developer e iniciamos “Alexa Skills Kit”
  + “Add a New Skill” -> creamos una nueva Skill, rellenando sólo el nombre de la Skill y su “invocation name”.

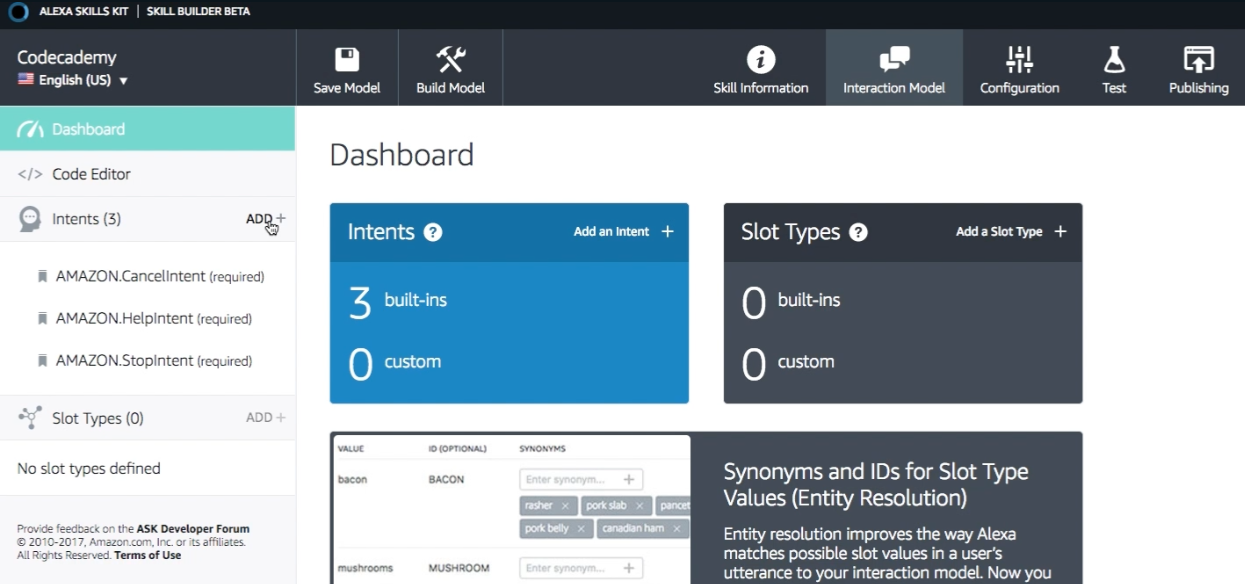
****

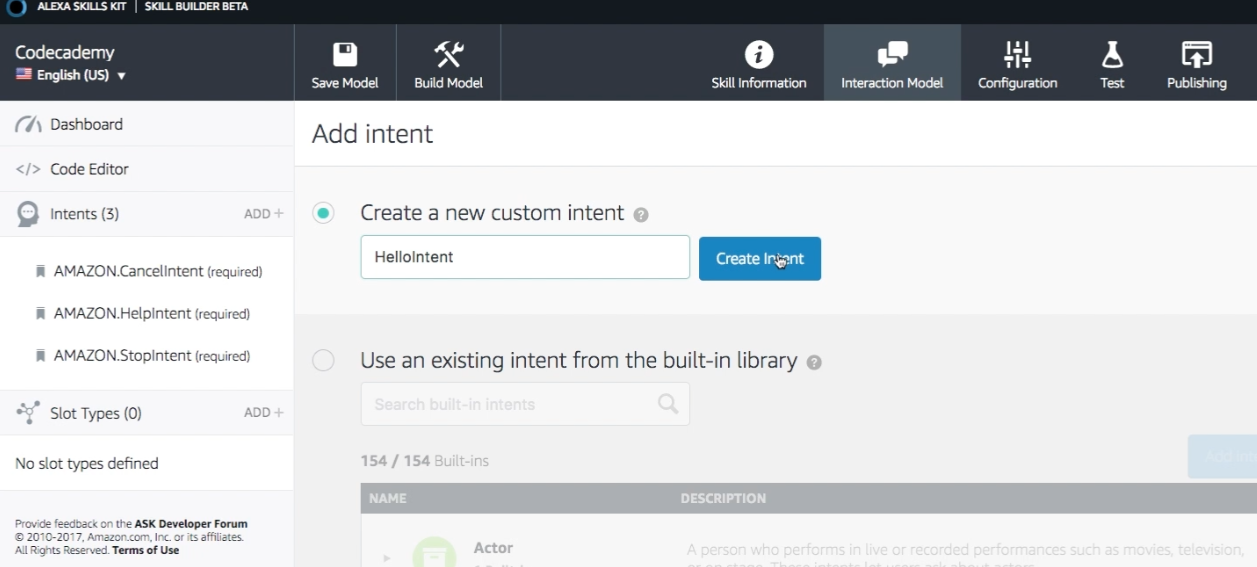
****

* + A continuación, en “Interaction Model” de la barra de menú, clickamos en “Launck Skill Builder” -> abriéndose la interfaz de “Alexa Skills Kit”

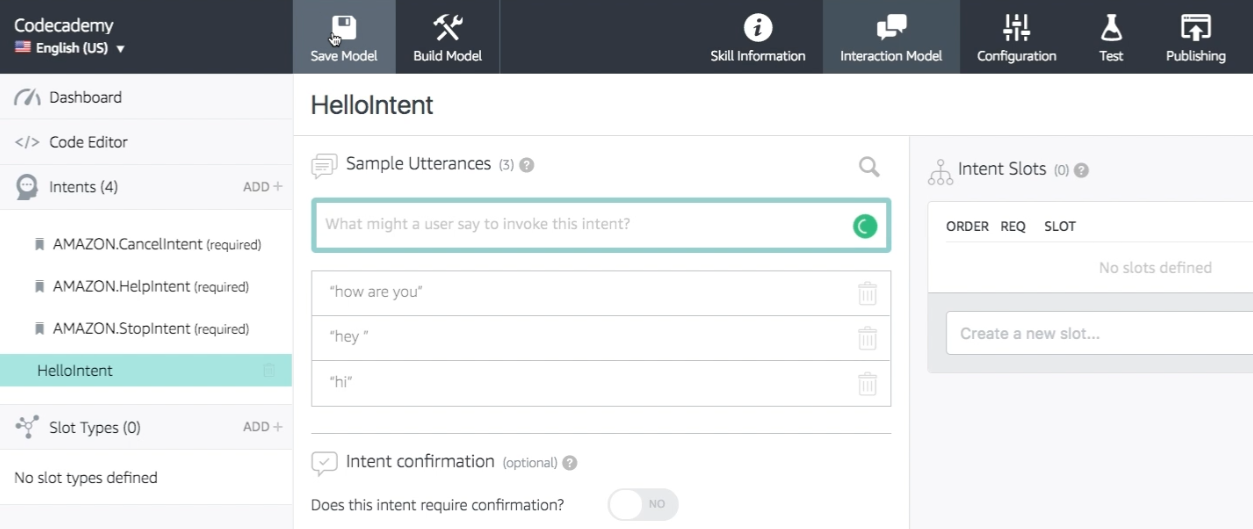
****

* + Una vez en la interfaz de “Alexa Skills Kit”, seleccionamos en “Add Intents” (barra de menú) y creamos el “Intent” y sus “Utterances”.

****

****

* + A continuación, clickamos, en la barra principal de menús, en “Save Model” y acto seguido en “Build Model”.

****

Ya tenemos contruído la Skill pero falta conectarla a su función lambda -> backend.

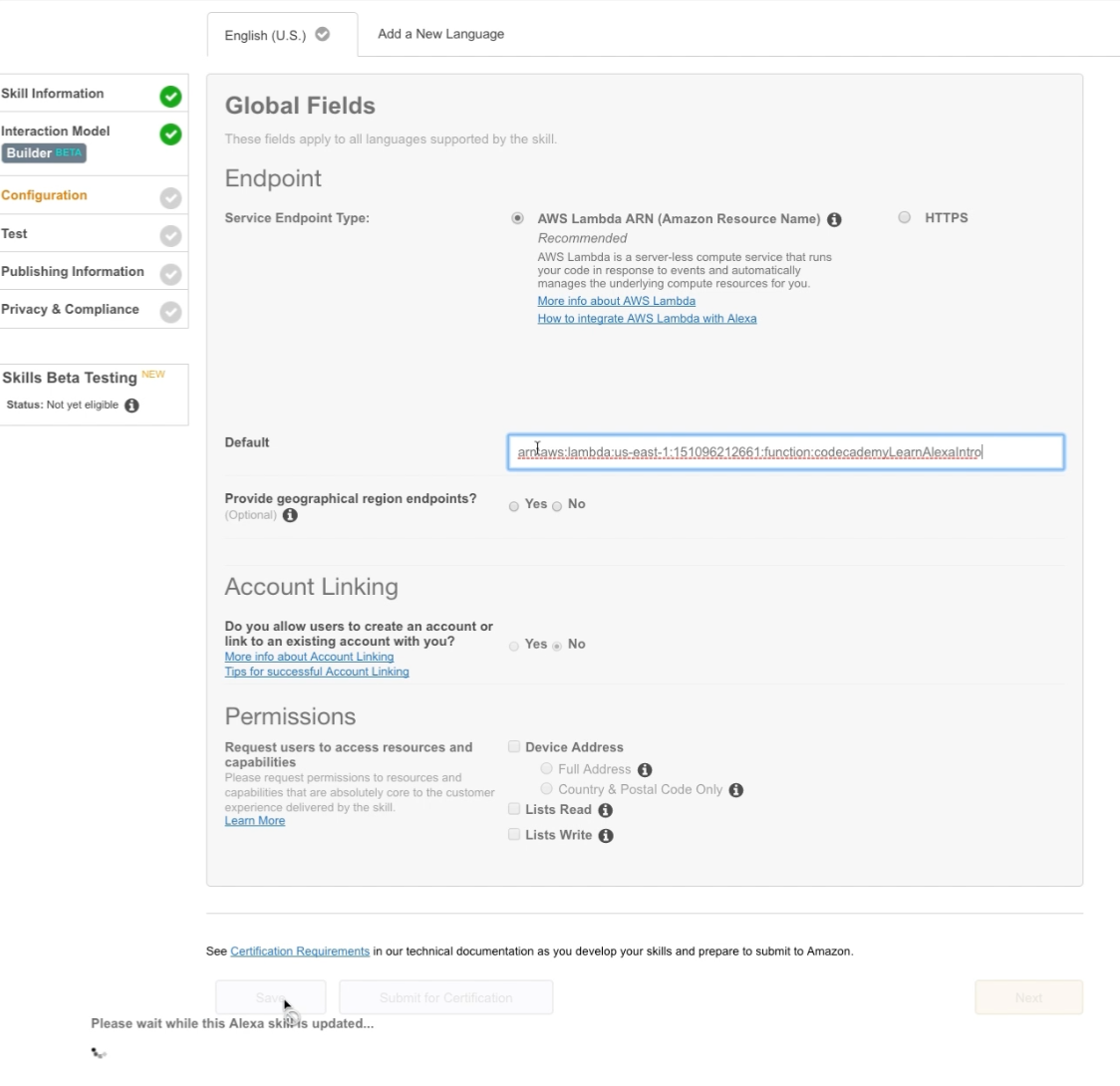
* **BACK-END** -> Una vez que hemos creado la iteracción del modelo (front-end), es hora de conectarlo a nuestra lógica del programa (back-end).

Usaremos **AWS Lambda Functions** para albergar la lógica del programa. Para conectar el front-end a una función lambda (back-end), tenemos que proporcionarle su Amazon Resource Name (ARN), que es un identificador único que representa nuestra lambda function.

Pasos para crear la lambda function

1. Clickamos en la barra de menú en la opción de “Configuration”.
2. Ahí, rellenamos las opciones disponibles de la siguiente manera:
   1. “EndPoint”: AWS Lambda ARN
   2. Metemos el código ARN de la función lambda que hemos programado.
   3. Clickamos en “Save” y ya tenemos nuestra Skill conectada y lista para su uso.

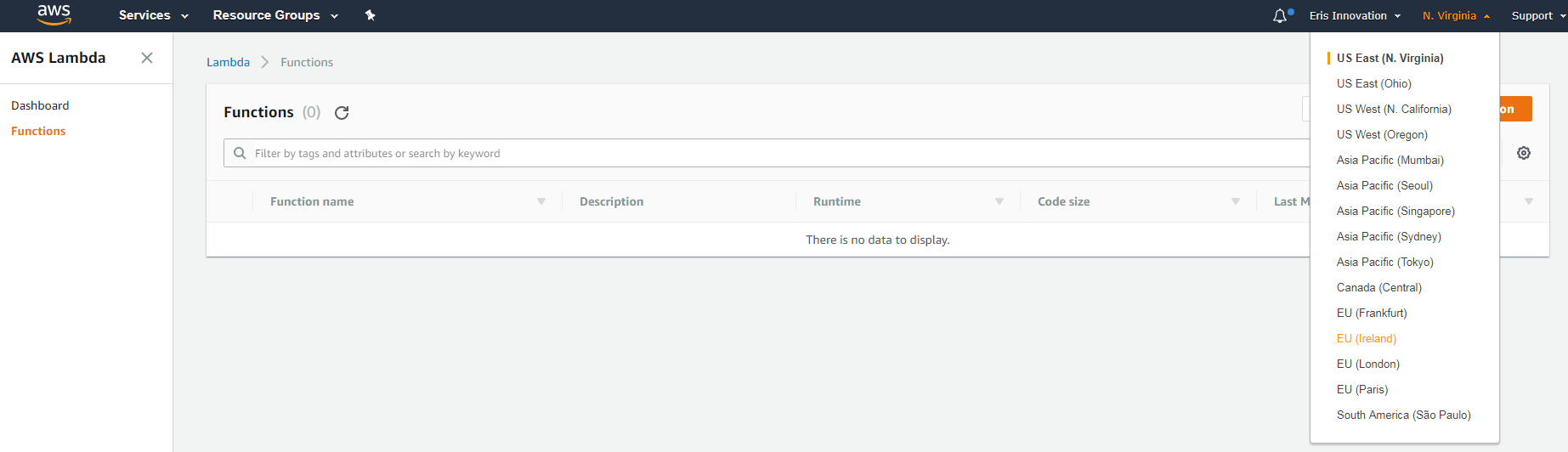
Podemos probar la Skill con el “Test Simulator”, clickando en la barra de menú en Test. O con “Echosim.io” (<https://beta.echosim.io/embed/welcome>).



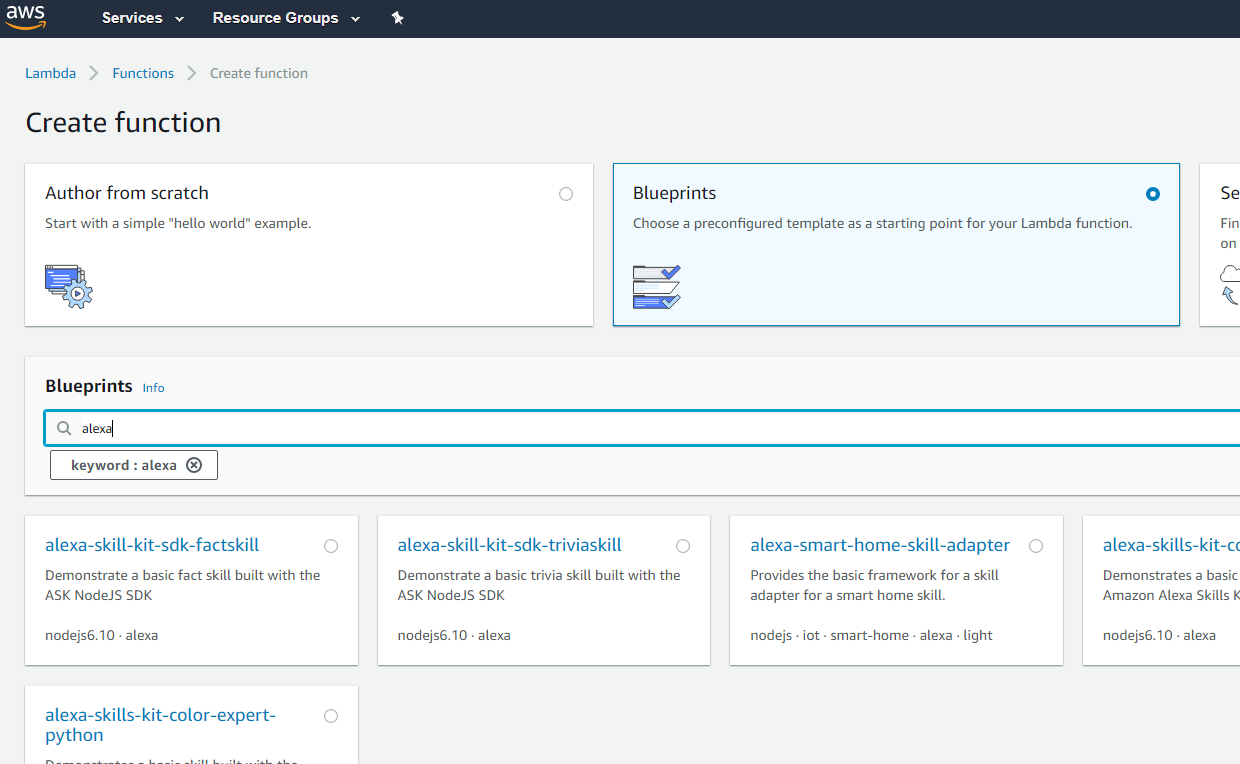
Pero, ¿cómo creamos una función lambda?

## Lambda Function

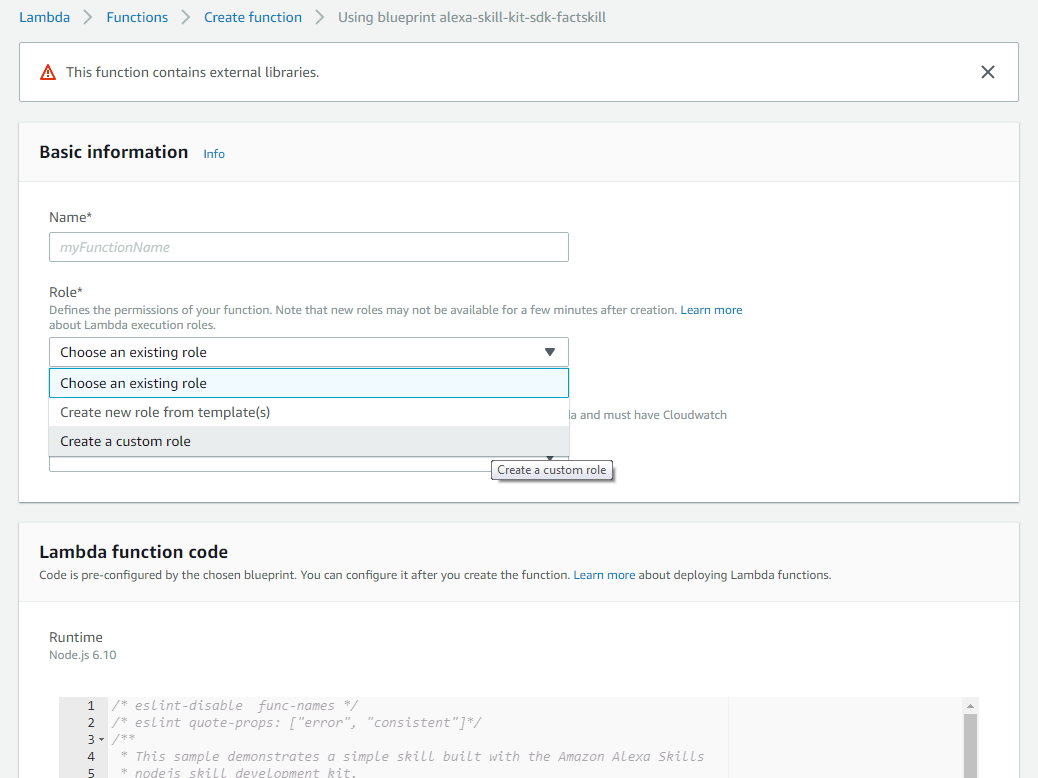
1. Crear cuenta/registrarse en AWS.
2. Buscar y seleccionar la opción de “Lambda” en los servicios disponibles (en la consola de AWS) -> Crear función.
3. Ajustamos la “AWS Region” -> ya que AWS Lambda sólo está disponible con Alexa Skills Kit en dos regiones “US East (N.Virginia)” y “EU (Ireland)”. Y seleccionamos la opción de “Create Function”.



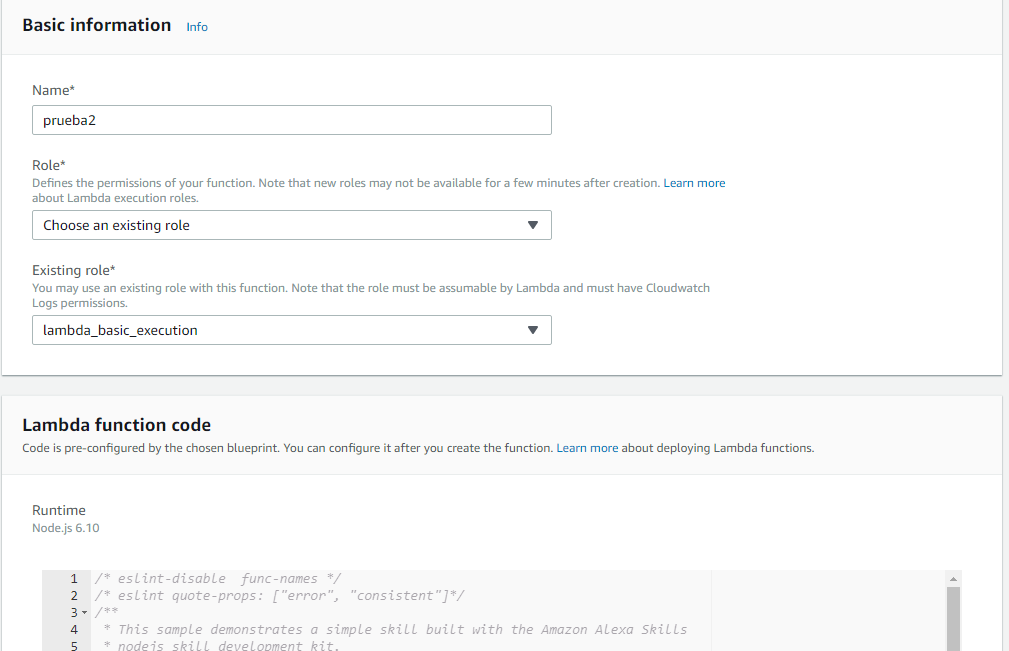
1. Crear la función: Para crear la función necesitamos el SDK de Alexa a nuestra función Lambda, y el “Blueprint” nos lo permite hacer automáticamente (buscamos y clickamos en “alexa-skill-kit-sdk-factskill”)



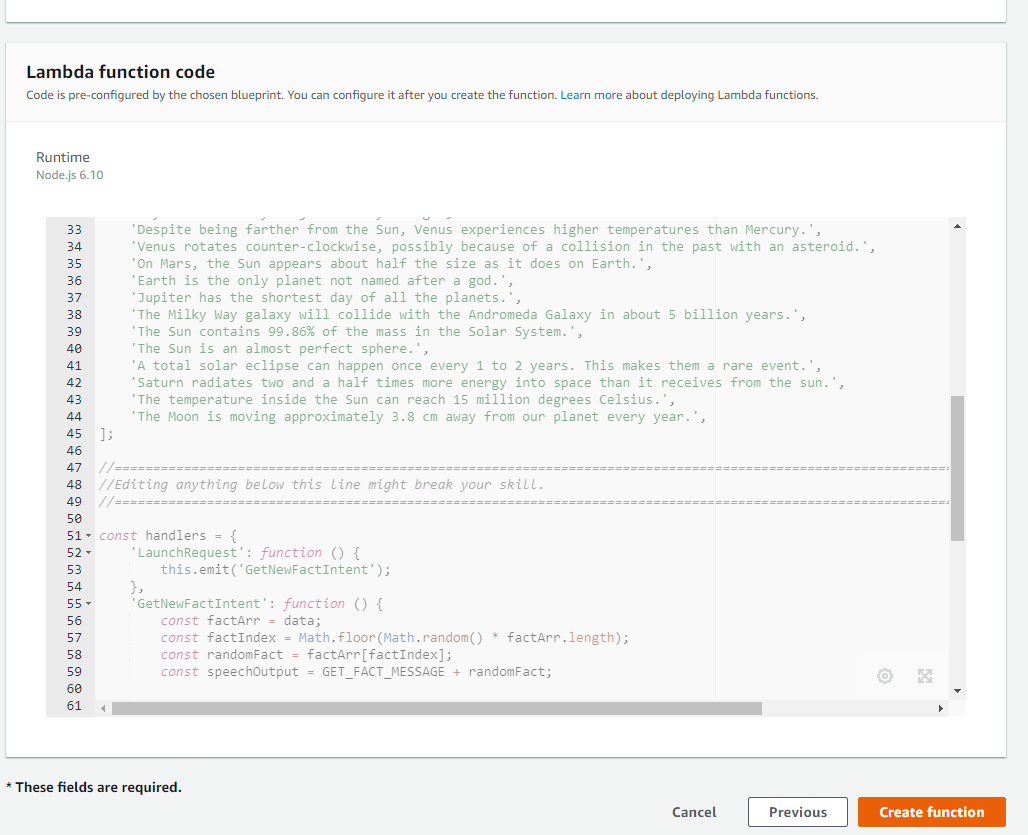
1. Se nos abrirá automáticamente la siguiente ventana donde escribiremos el nombre de la función y en “Role” seleccionaremos la opción de “Create a custom role”.



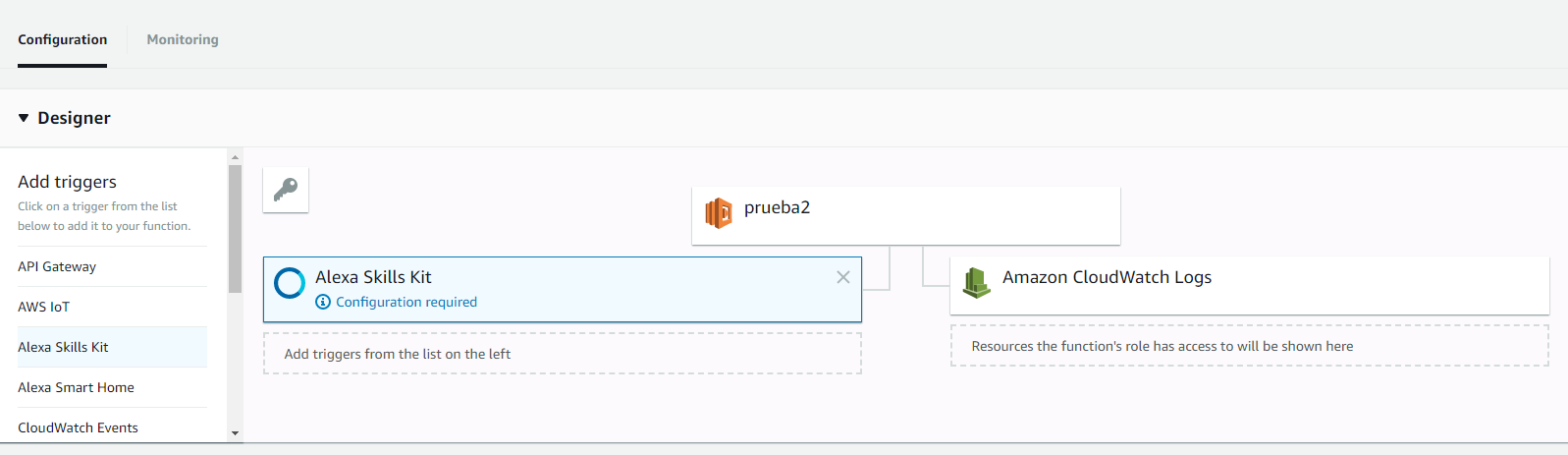
1. A continuación, nos aparecerá la siguiente ventana y clickaremos en “allow”. Obteniendo:



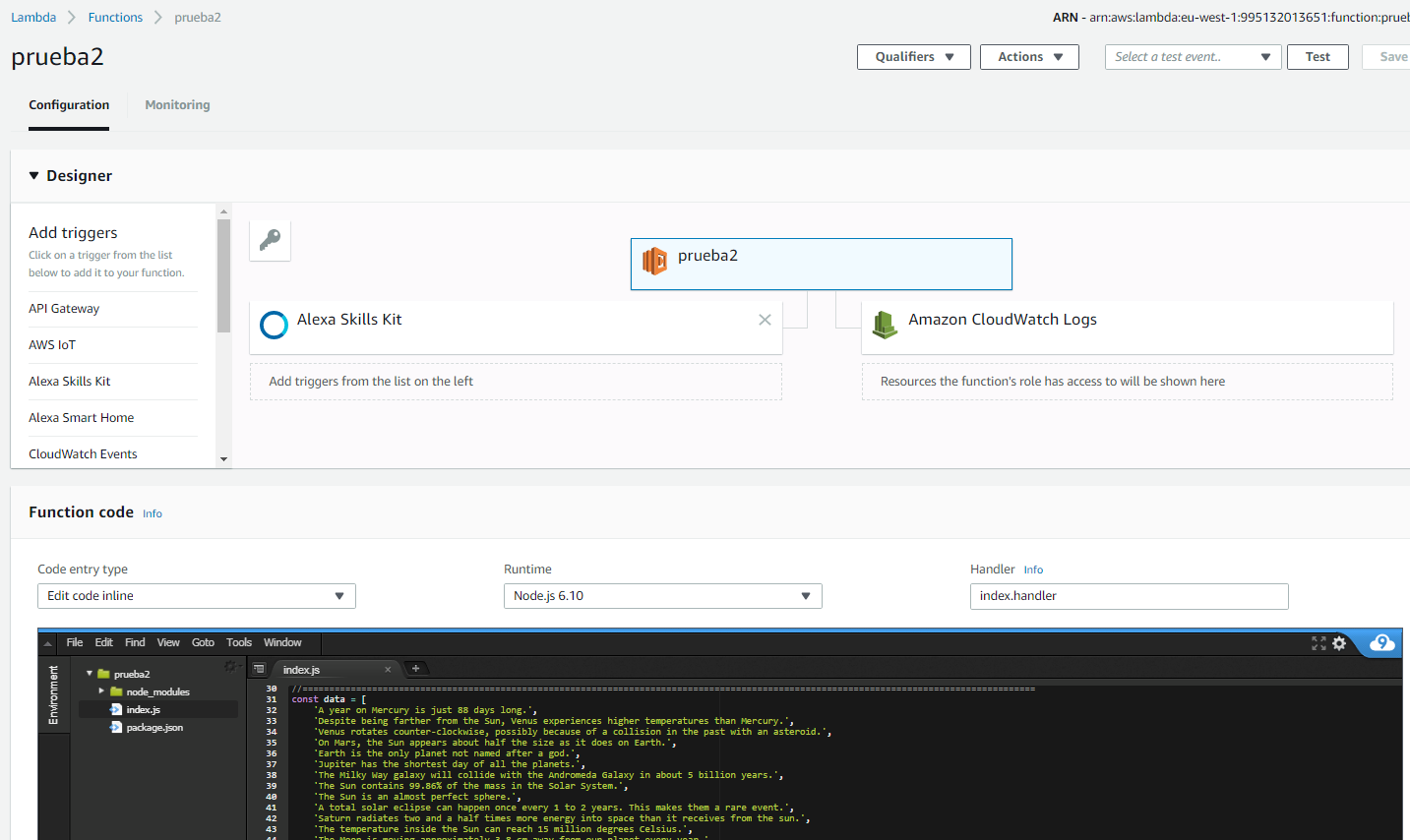
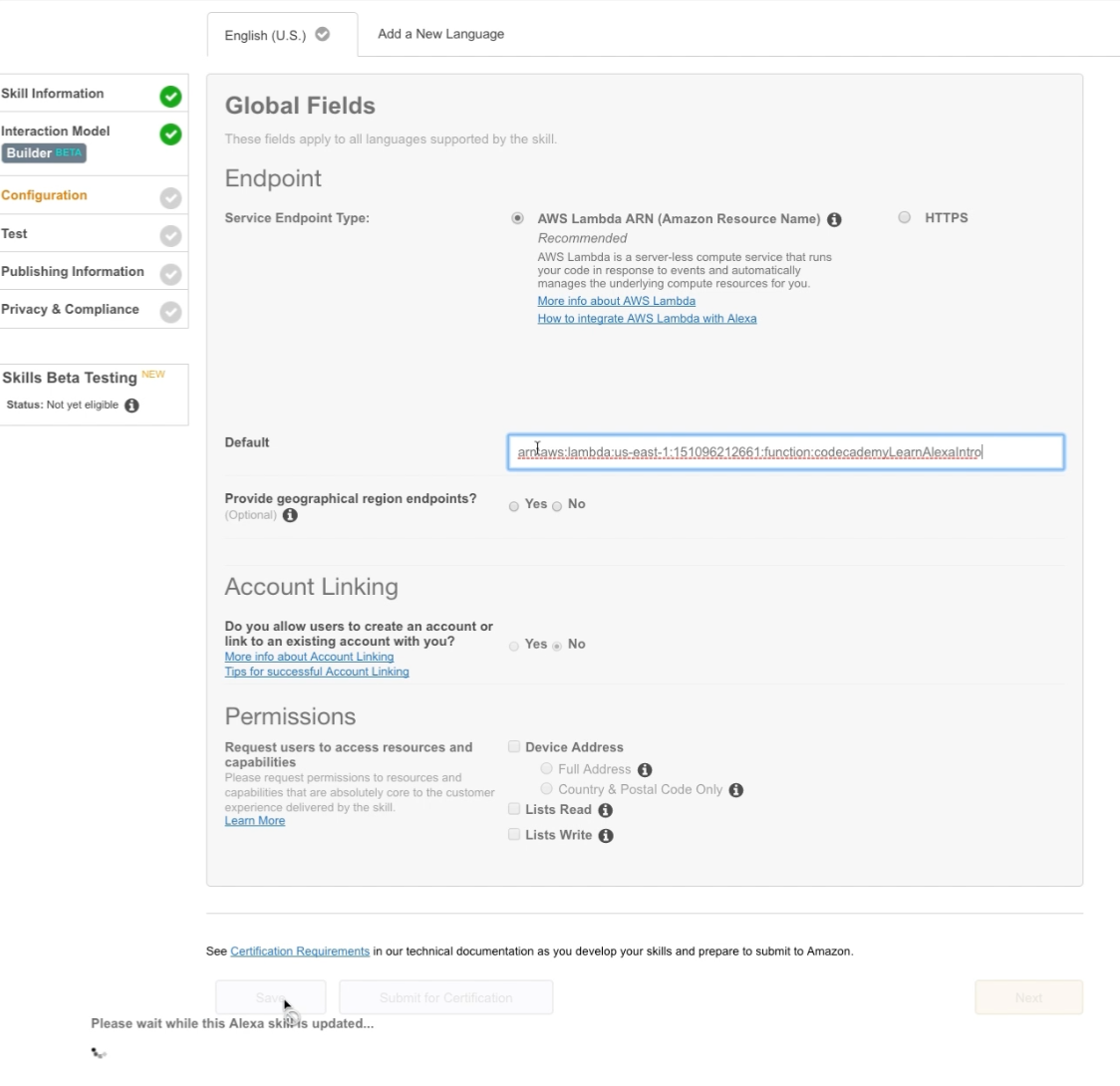
1. Clickamos en “Create Function”.



1. Añadimos el trigger de “Alexa”.

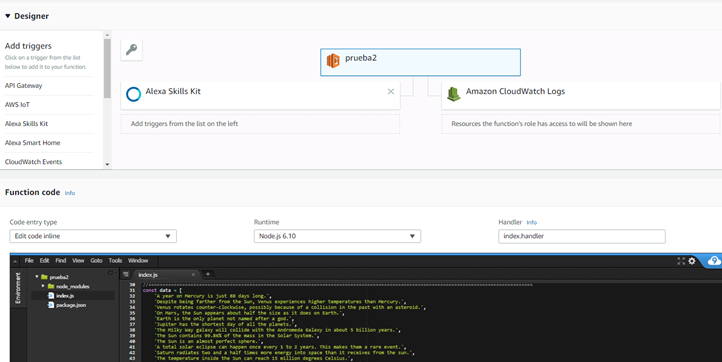
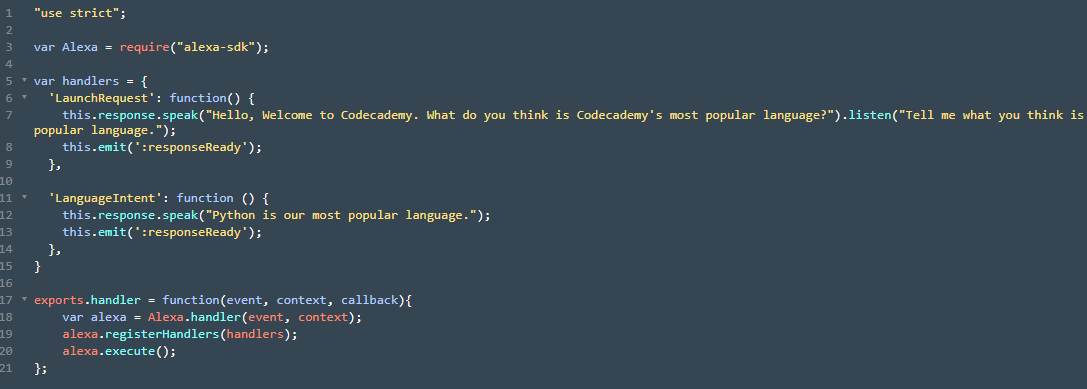


1. En el apartado “Function Code” es donde modificaremos el código de la función (parte negra) y el código ARN está en la esquina superior derecha (que deberemos meter en el EndPoint de la Skill).



## Programar una Lambda Function

Vamos a programar en el archivo “index.js” de la ventana de “Lamba Function”.



A continuación, explicamos cada parte del código:

1. **Var Alexa = require(“Alexa-sdk”)** -> Incluye Alexa-SDK library en nuestra función lambda.
2. **Var handlers = {}** -> Cuando Alexa manda una petición a la función lambda, el script necesita manejar/gestionar las peticiones. El objeto “handlers” le dice a Alexa cómo gestionar varias acciones.

Alexa SDK library incluye los métodos “this.response.speak()” (genera la respuesta de Alexa) y “this.emit()” (envía la respuesta de Alexa al usuario) que generan de forma conjunta un JSON de respuesta de la función.

Sin embargo, “this.response.speak()” genera una respuesta y luego hace que Alexa cierre la sesión. Cuando se crea una conversación, normalmente no se quiere que Alexa cierre sesión. Alexa SDK’s “.listen()” method nos permite mantener la sesión abierta para permitir un diálogo:

**this.response.speak(“what do you think is it the most popular language?”).listen(“Sorry, I don’t understand”); -> si el usuario no contexta o Alexa no entiende la respuesta, el dispositivo retoma la conversación con la frase “Sorry, I don’t understand”.**

1. **“LaunchRequest”: function () {this.response.speak(…); this.emit(…)}** -> cuando el usuario invoca la Skill con el “invocation name”, pero no da ningún “intent”.

**Ejemplo**:

Usuario: “Open CodeAcademy” ≠ Ask to CodeAcademy to say Hello (“intent”)

Alexa (LaunchRequest): “Welcome to CodeAcademy” ≠ No hay LaunchRequest

1. Una vez que temenos nuestros controladores (“handlers”), nos queda un paso para completar nuestra función lambda: añadir una función que AWS invoque cuando Alexa ejecute el código:

**Exports.handler = function (event, context, callback) {**

**//Set up the Alexa object**

**Var alexa = Alexa.handler(event, context);**

**//Register handlers**

**alexa.registerHandler(handlers);**

**//Start our Alexa code**

**alexa.execute();**

**}**

## Interactuar con las respuestas de los usuarios (slots)

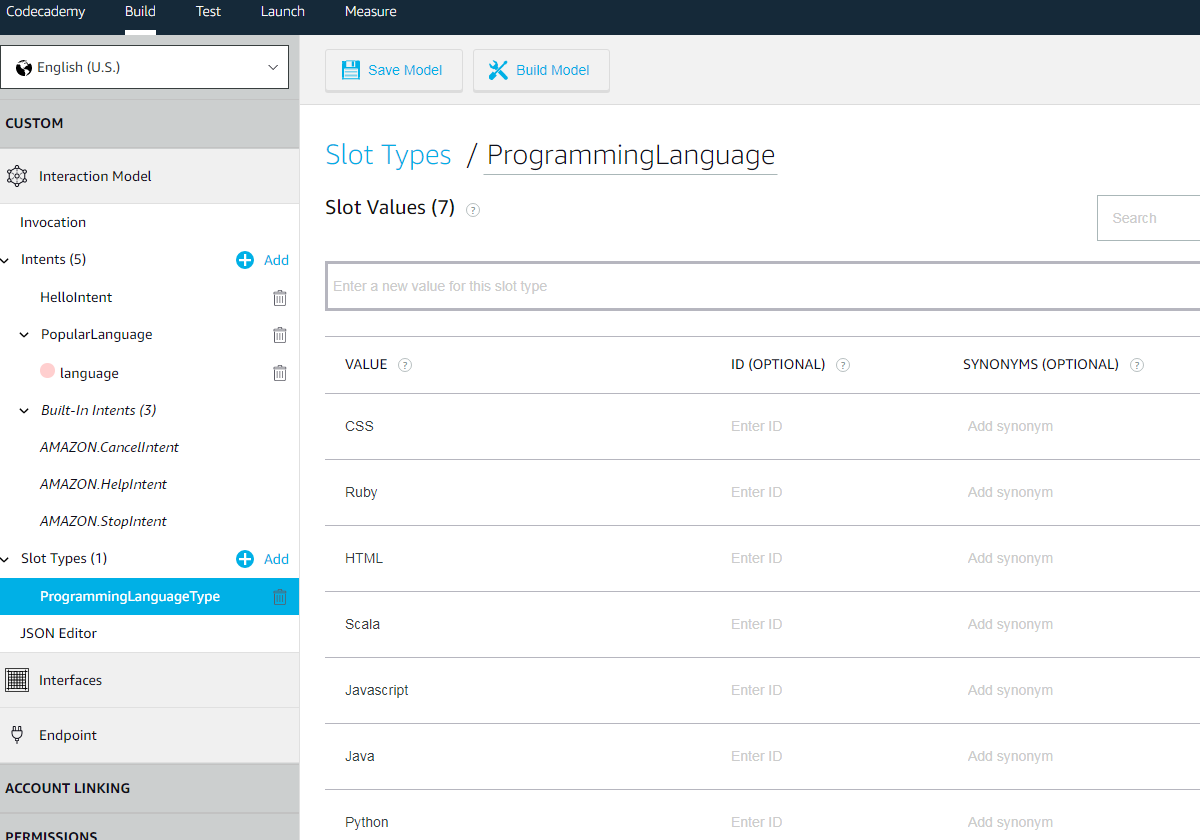
Un “slot” es un argumento dado por el usuario en un “intent” que porporciona a Alexa más información acerca de la petición. -> La respuesta del usuario es capturada y evaluada por Alexa.

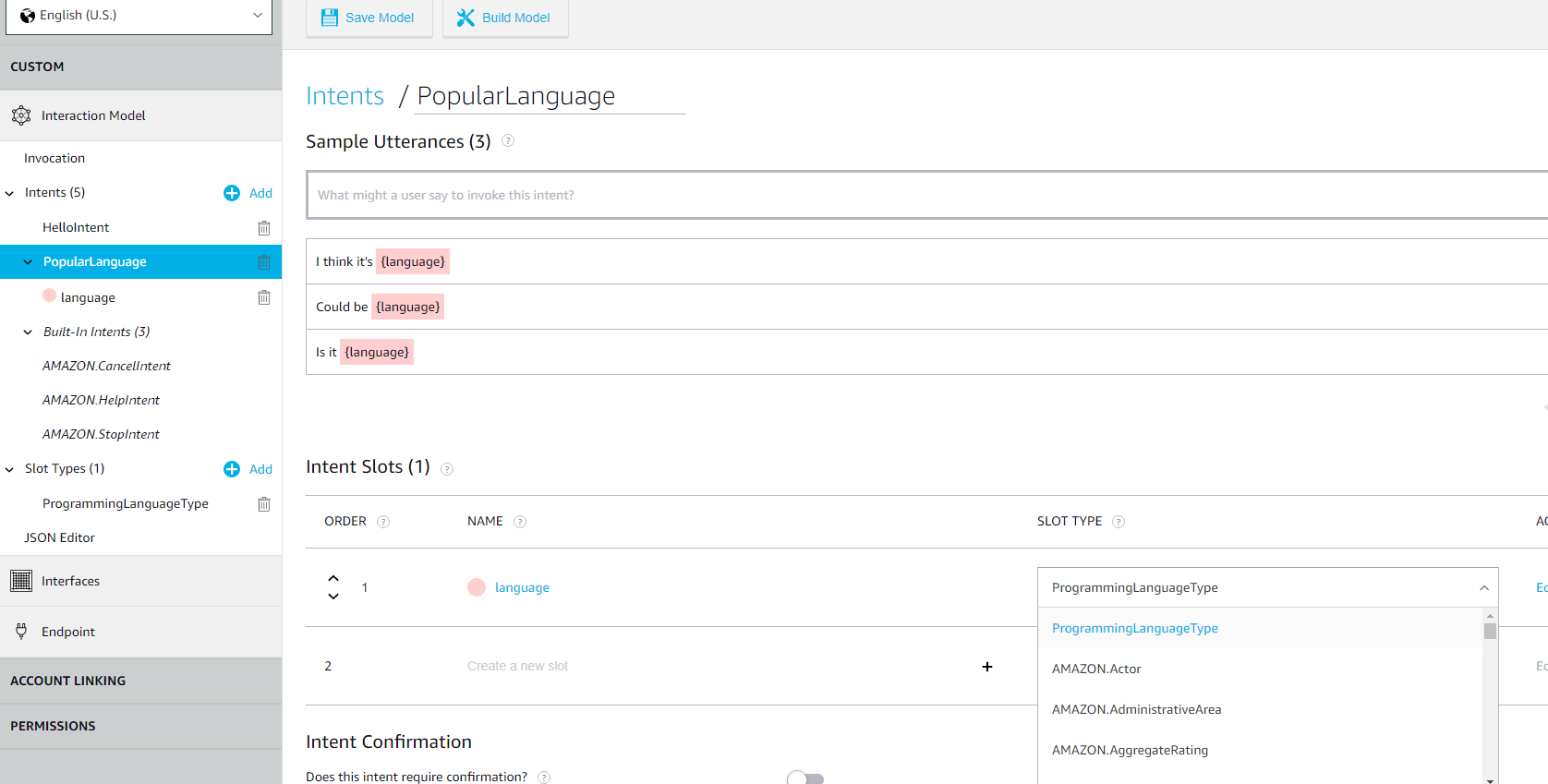
Ejemplo:

“Alexa, ask History Buff what happened on June third” -> “June third” is the value of a date slot that interprets the request.

Usaremos “slots” para capturar inputs del usuario y actualizar nuestra función lambda para evaluarlos y responder en base al input recibido.

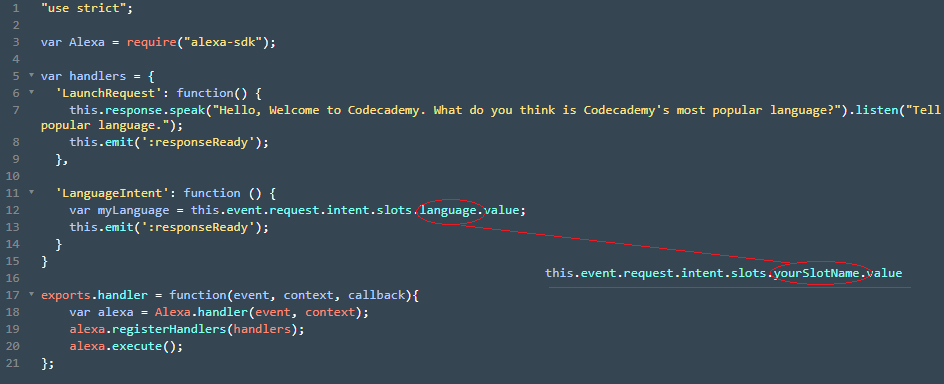
1. En la interfaz de Alexa Developer añadimos un nuevo “slot” (“Slot Types” -> Add) e introducimos los valores que podrá tomar.



1. Creamos un nuevo “Intent” e introducimos las posibles “utterances” del usuario, sustituyendo la respuesta por el nombre de la variable {language}
2. A continuación, asignamos a esa variable una tipología de slot:

Los slots son enviados (petición) en un archivo JSON desde Alexa hasta la función. Necesitaremos, por tanto, modificar nuestra función lambda para leer los valores de los slots.

En Alexa-SDK, el JSON de petición es disponible como un objeto. Podemos acceder al objeto usando “this.event.request.intent.slots.YourSlotName.value”.



## Session Attributes

Session attributes allow Alexa to remember and update user-provided details throughout a session.

You can think of session attributes as being similar to variables that store information about the program, except values stored in session attributes persist throughout a session. You can persist attributes between sessions by writing them to a DynamoDB database (aka long-term memory).

Session attributes are stored in the response that your skill receives from Alexa as a JSON. We can create and edit attribute fields by accessing their key. Inside of the Lambda function, you can access session attribute values using this.attributes.

In the example above, we create a field in attributes and save the string 'value' to it. this references the alexa object that's instantiated in the export.handlers function.



Since the LaunchRequest is usually the first intent that the skill executes, we often set the initial attribute state inside of it.





La siguiente imagen muestra el código utilizado para crear una función lambda que permita a Alexa preguntar acerca de un determinado lenguaje de programación (fijado por el usuario) y guardar la sesión y los resultados:

