

# **Tutorial de desenvolvimento de uma skill Alexa**

**Rafael dos Santos Rodrigues, Dalton Solano dos Reis** – Orientador

Curso de Bacharelem Sistemas de Informação

Departamento de Sistemas e Computação

Universidade Regional de Blumenau (FURB) – Blumenau, SC – Brazil

## **Objetivo:**

Este documento tem como objetivo apresentar alguns conceitos ao desenvolver uma skill para Alexa, como criar uma skill para a Alexa, como criar e hospedar um servidor na AWS Lambda, criar uma “coisa” e vincular ao ESP32 e utilizar alguns serviços auxiliares sendo eles: DynamoDB e CloudWatch.

## **Configurando Arduino IDE:**

O Arduino IDE é um ambiente de desenvolvimento de código aberto para programação de microcontroladores. Ele está em constante desenvolvimento e pode suportar um número crescente de plataformas, incluindo a maioria dos módulos baseados em ESP32. Ele deve ser instalado junto com as definições da placa ESP32, biblioteca MQTT e biblioteca ArduinoJson.

O Arduino IDE pode ser baixado no site do Arduino:

<https://www.arduino.cc/en/Main/Software>

Após a instalação, abra o Arduino IDE e siga a sequência e as figuras abaixo:

Passo 1 - Para configurar o ESP32 no Arduino IDE, entre em "File" -> "Preferences";

Passo 2 - E cole o seguinte URL no campo "Additional Boards Manager URLs":

[https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

Obs.: Caso altere o idioma é necessário reiniciar o Arduino IDE.

Passo 3 - Entre em "Tools" -> "Board" -> "Boards Manager" procure por "esp32" e instale a última versão da "esp32 by Espressif Systems" da lista, nesse processo o Arduino IDE irá baixar os arquivos necessários para o ESP32.

Ao abrir o "Boards Manager" ele pode demorar cerca de 20 segundos para atualizar todos os arquivos de hardware (se a rede estiver em más condições, pode levar mais tempo).

Passo 4 - Abrir novamente o "Tools" e validar se o item "Board" se encontra com o modelo do seu componente, caso não alterar e selecionar o correto.

É possível conferir os modelos de placas compatíveis com o Arduino IDE pelo documento disponibilizado pela "Espressif Systems" no site:

<https://github.com/espressif/arduino-esp32/blob/master/boards.txt>

Passo 5 - Abra o item "Sketch" -> "Include Library" -> "Manage Libraries" procure por "MQTT" e instale a última versão do criado "Joel Gaehwiler".

Passo 6 - Abra o item "Sketch" -> "Include Library" -> "Manage Libraries" procure por "ArduinoJson" e instale a última versão do criado "Benoit Blanchon".

Figura 1 - Passo 1 -> Configurando Arduino para o ESP32

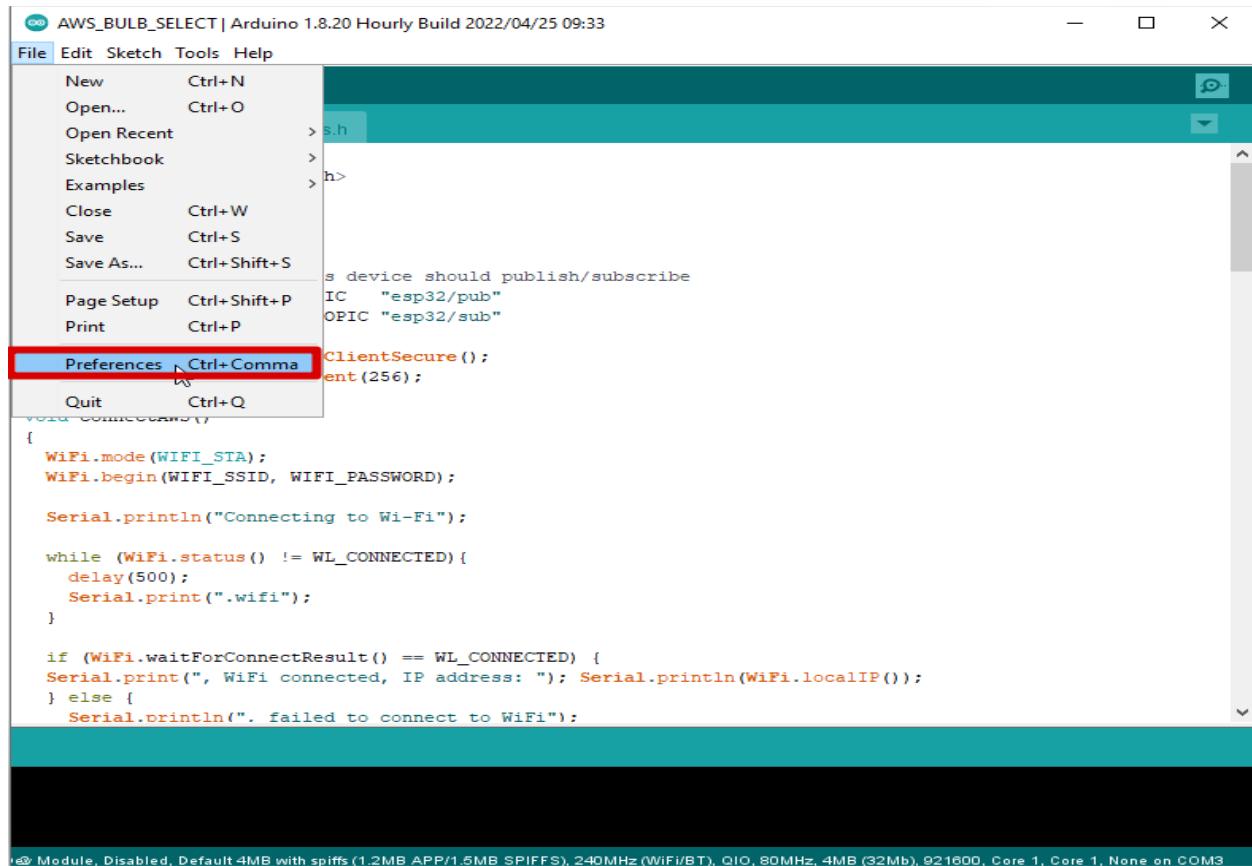


Figura 2 - Passo 2 -> Configurando Arduino para o ESP32 (continuação)

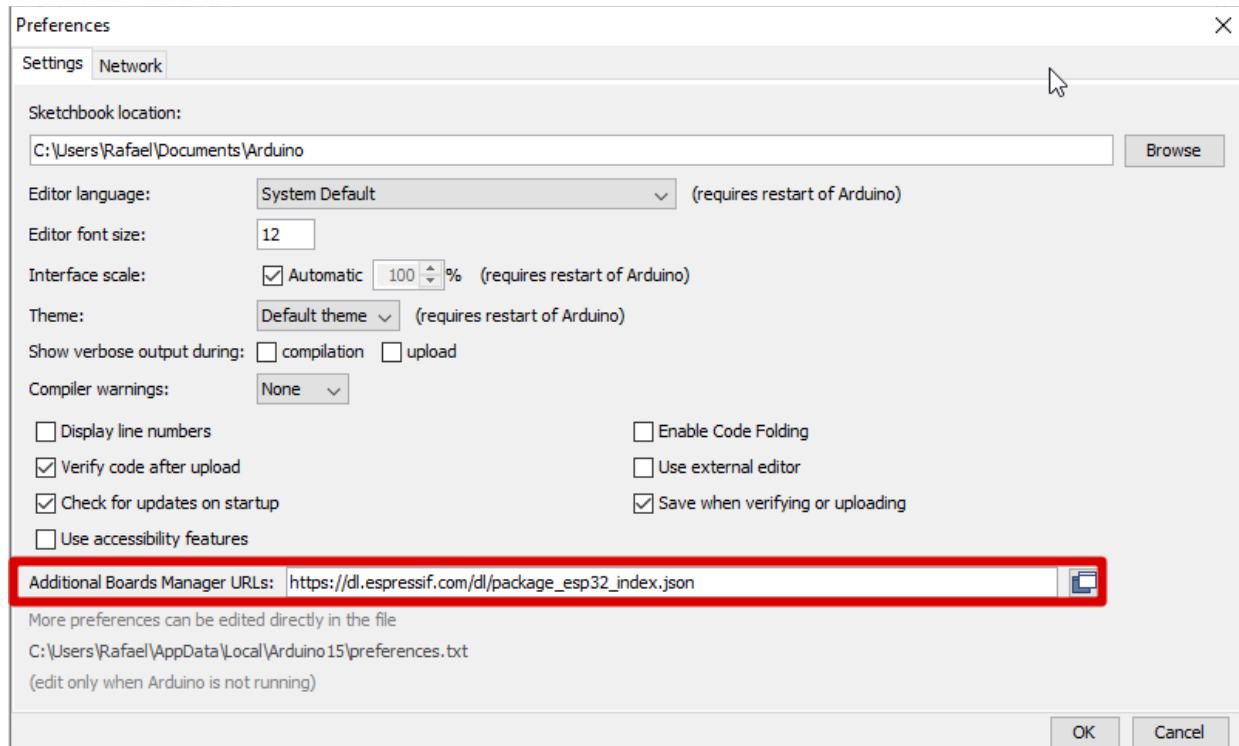


Figura 3 - Passo 3 -> Instalando pacote ESP32

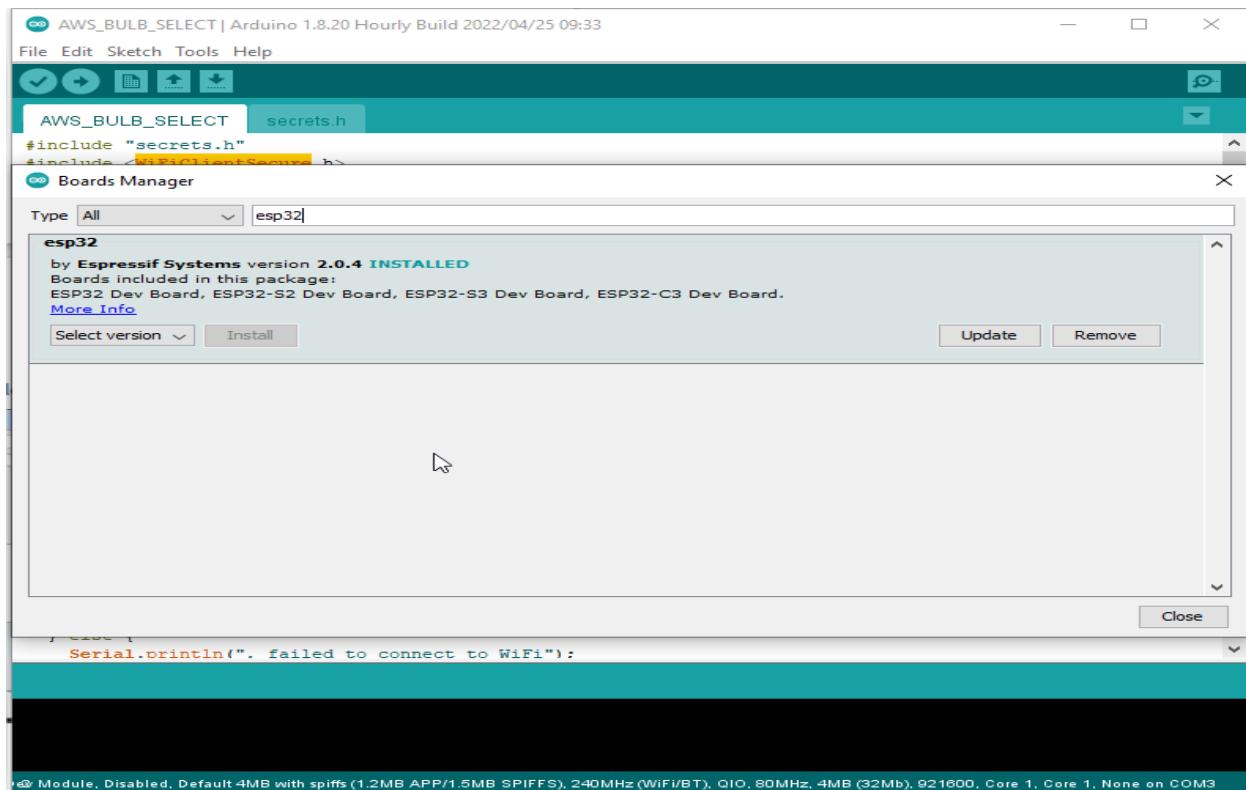


Figura 4 - Passo 4 -> Selecionando placa ESP32

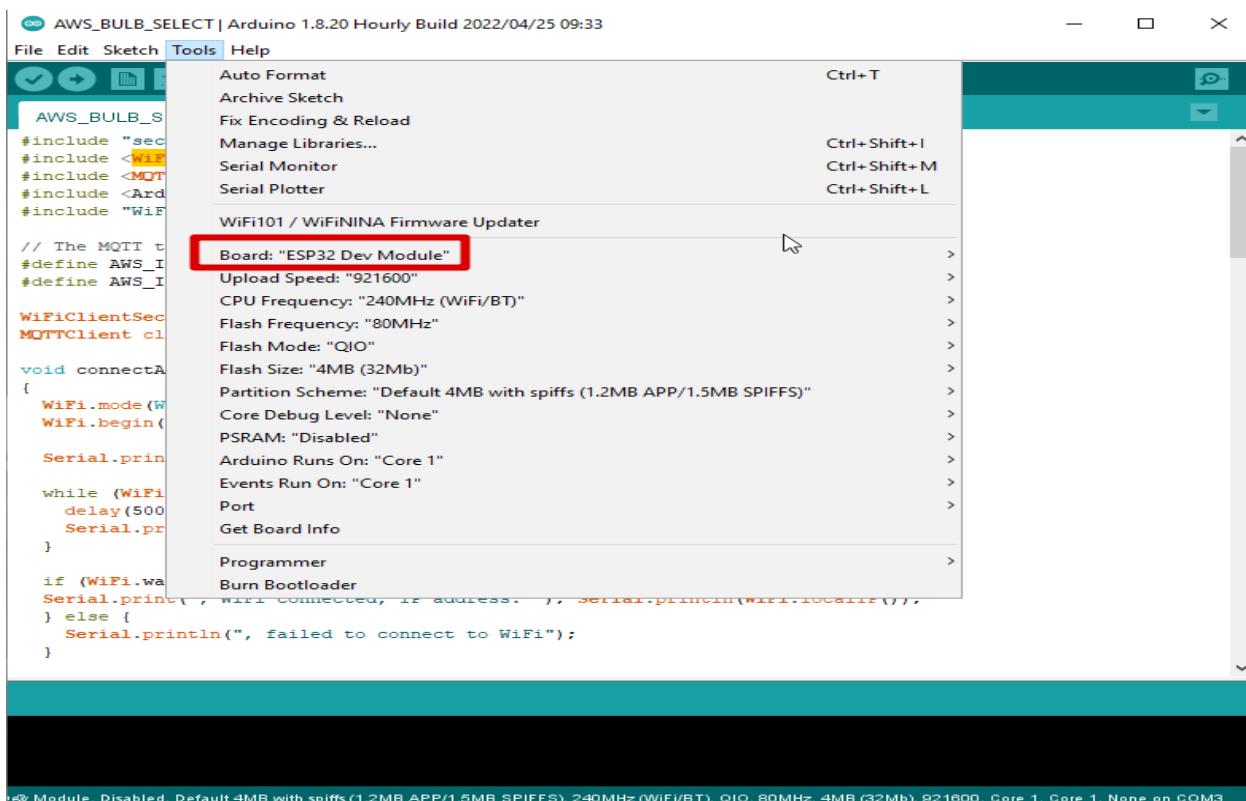


Figura 5 - Passo 5 -> Instalando biblioteca MQTT

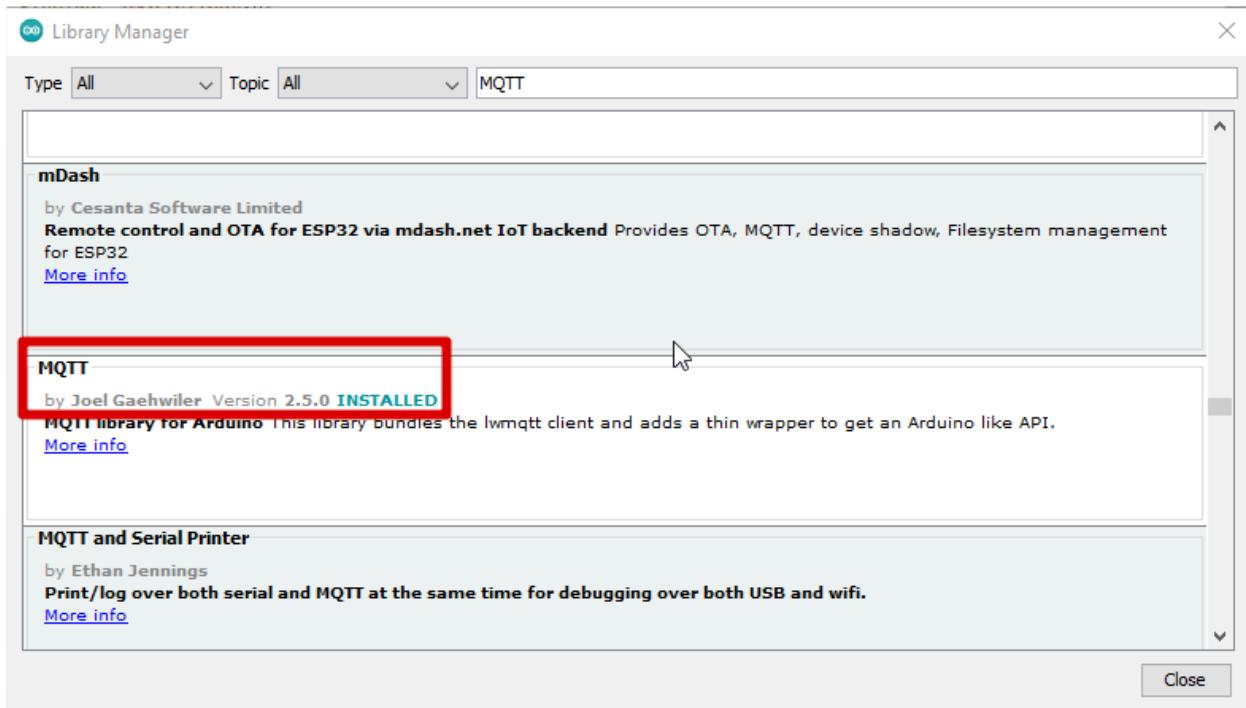
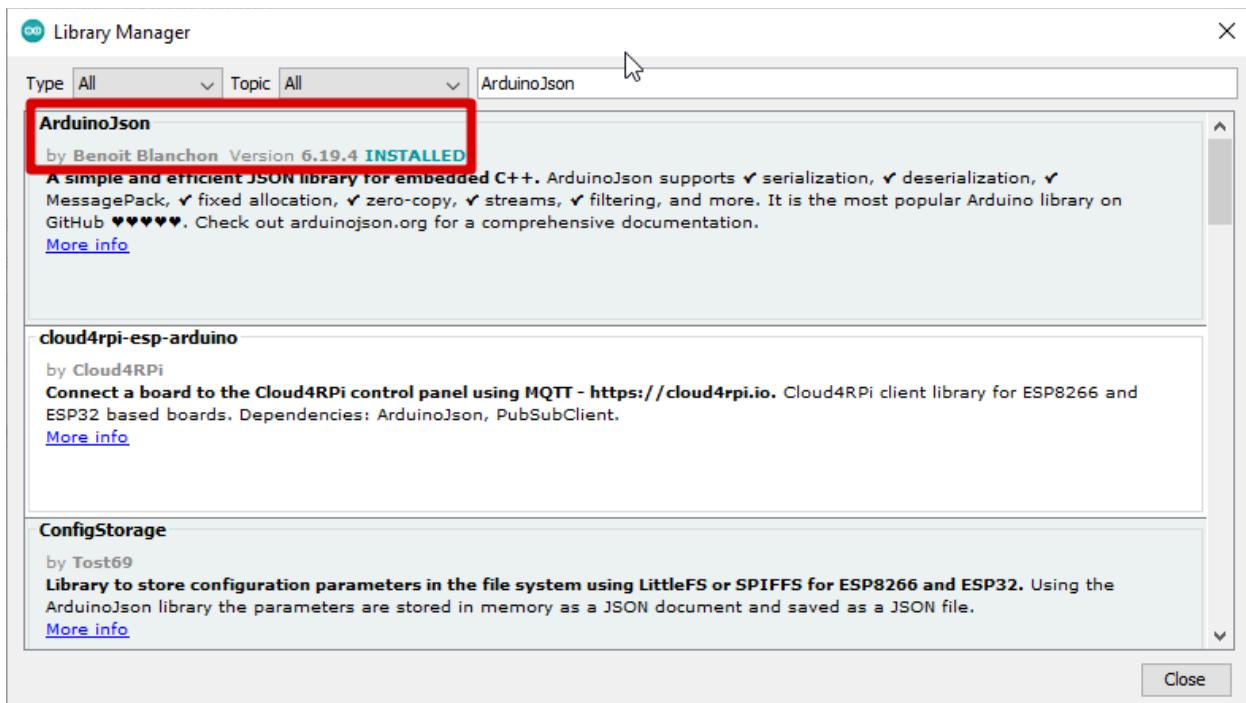


Figura 6 - Passo 6 -> Instalando biblioteca ArduinoJson



#### Código Arduino IDE

O código a seguir se encontra no item “Código Arduino” que foi disponibilizado no github pelo link:  
<https://github.com/rafsarodrigues/Alexa-Tutorial-PT-BR/>

Será quebrado o código explicando para que serve cada parte e o que é necessário ser alterado para executar com os seus dados.

### Secrets.h

No arquivo “secrets.h” ele serve como auxiliar para armazenar os certificados, chaves e dados do Wi-Fi que será utilizado pelo ESP32.

Na variável “THINGNAME” seria o nome da “coisa” que será criada posteriormente na Amazon que será o ponto de acesso entre o ESP32 e o AWS Lambda.

```
#define THINGNAME "ESP32_FURB_G"
```

Nas constantes WIFI\_SSID, WIFI\_PASSWORD e AWS\_IOT\_ENDPOINT, respectivamente são o login e senha do Wi-Fi e o endpoint disponibilizado no AWS Lambda.

```
const char WIFI_SSID[] = "";
const char WIFI_PASSWORD[] = "";
const char AWS_IOT_ENDPOINT[] = "a1tisats.iot.sa-east-1.amazonaws.com";
```

As constantes AWS\_CERT\_CA, AWS\_CERT\_CRT e AWS\_CERT\_PRIVATE são também informações que terão que ser preenchidas conforme “coisa” criada na Amazon sendo respectivamente a Amazon Root CA 1, Device Certificate e Device Private Key.

```
static const char AWS_CERT_CA[] PROGMEM = R"EOF()EOF";
static const char AWS_CERT_CRT[] PROGMEM = R"KEY(KEY";
static const char AWS_CERT_PRIVATE[] PROGMEM = R"KEY(KEY";
```

Essas informações são sensíveis pois dependendo do tipo de política utilizada na Amazon qualquer pessoa com esses dados poderá acessar o seu endpoint e executar requisições.

### AWS\_BULB\_SELECT.ino

O arquivo “AWS\_BULB\_SELECT.ino” contém o código principal onde será realizada o recebimento e envio das informações para a Amazon AWS e realizada a conexão com o Wi-Fi.

Primeiro é realizado a importação do arquivo “secrets.h” e as bibliotecas importadas que serão utilizadas no código.

```
#include "secrets.h"
#include <WiFiClientSecure.h>
#include <MQTTClient.h>
#include <ArduinoJson.h>
#include "WiFi.h"
```

Definimos em constantes os tópicos MQTT que iremos utilizar para publicar e se inscrever.

```
#define AWS_IOT_PUBLISH_TOPIC "esp32/pub"
#define AWS_IOT_SUBSCRIBE_TOPIC "esp32/sub"
```

Com isso a ordem de execução primeiro será o método “setup()” que executa o “connectAWS()” que irá se conectar ao Wi-Fi e após sucesso se inscrever no tópico criado na Amazon AWS. Posteriormente, será executado o método “setupBulb()” que irá escolher quais pinos do ESP32 serão utilizados.

```
void setup() {
    Serial.begin(9600);
    connectAWS();
    setupBulb();
}
```

```
void connectAWS()
{
    WiFi.mode(WIFI_STA);
    WiFi.begin(WIFI_SSID, WIFI_PASSWORD);

    Serial.println("Connecting to Wi-Fi");

    while (WiFi.status() != WL_CONNECTED){
```

```

delay(500);
Serial.print(".wifi");
}

if (WiFi.waitForConnectResult() == WL_CONNECTED) {
Serial.print(", WiFi connected, IP address: "); Serial.println(WiFi.localIP());
} else {
Serial.println(", failed to connect to WiFi");
}

// Configure WiFiClientSecure to use the AWS IoT device credentials
net.setCACert(AWS_CERT_CA);
net.setCertificate(AWS_CERT_CRT);
net.setPrivateKey(AWS_CERT_PRIVATE);
// Connect to the MQTT broker on the AWS endpoint we defined earlier
client.begin(AWS_IOT_ENDPOINT, 8883, net);
// Create a message handler
client.onMessage(messageHandler);

Serial.print("Connecting to AWS IOT");

while (!client.connect(THINGNAME)) {
Serial.print(".");
delay(100);
}

if(!client.connected()){
Serial.println("AWS IoT Timeout!");
return;
}
// Subscribe to a topic
client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
Serial.println("AWS IoT Connected!");
}

void setupBulb()
{
pinMode(14, OUTPUT);
pinMode(25, OUTPUT);
pinMode(26, OUTPUT);
pinMode(27, OUTPUT);
}

```

Com isso será executado em loop o método “loop()” que irá validar a conexão com a rede e Amazon AWS e caso algum deles caia o método “reconnect()” irá tentar reconectar com esses serviços.

```

void loop() {
if (!client.connected()) {reconnect();}
client.loop();
delay(1000);
}

```

```

void reconnect() {
// Loop until we're reconnected
while (!client.connected()) {
Serial.print("Attempting MQTT connection...");
```

```

// Create a random client ID
String clientId = "ESP8266Client-";
clientId += String(random(0xffff), HEX);
// Attempt to connect
if (client.connect(THINGNAME)) {
    Serial.println("connected");
    // Once connected, publish an announcement...
    client.publish(AWS_IOT_PUBLISH_TOPIC, "Reconnected");
    // ... and resubscribe
    client.subscribe(AWS_IOT_SUBSCRIBE_TOPIC);
} else {
    Serial.print("failed, rc=");
    Serial.println(" try again in 5 seconds");
    // Wait 5 seconds before retrying
    delay(5000);
}
}
}

```

Ao receber um mensagem via tópico MQTT será chamado o método `messageHandler(String &topic, String &payload)` que recebe o nome do tópico e alguma “String”, neste caso será esperado um arquivo JSON. Esse método irá “deserializar” o JSON criando uma lista sendo que cada item será referenciado a uma variável para realizar as tratativas, neste caso seria gerar a ordem de execução das lâmpadas e por fim é executado o método “`publishReturnMessage()`” para retornar uma mensagem de sucesso para a Amazon AWS.

```

void messageHandler(String &topic, String &payload){
    Serial.println("Log incoming: " + topic + " - " + payload);

    StaticJsonDocument<200> doc;
    deserializeJson(doc, payload);
    String sequence = doc["sequence"];
    char str_array[sequence.length()];
    sequence.toCharArray(str_array, sequence.length());
    const String location = doc["location"];
    const int delayBulb = doc["delay"];
    byte index = 0;
    uint8_t strings[128]; // an array of pointers to the pieces of the above array after strtok()
    char *ptr = NULL;
    ptr = strtok(str_array, ",");
    while (ptr != NULL)
    {
        strings[index] = (uint8_t)atoi(ptr);
        index++;
        ptr = strtok(NULL, ",");
    }
    Serial.println("The Pieces separated by strtok()");
    for (int n = 0; n < index; n++)
    {
        Serial.print(n);
        Serial.print(" ");
        Serial.println(strings[n]);
        digitalWrite(strings[n], HIGH);
        delay(delayBulb);
        digitalWrite(strings[n], LOW);
        delay(100);
    }
    publishReturnMessage();
}

```

```
void publishReturnMessage()
{
    StaticJsonDocument<200> doc;
    doc["finish"] = "OK_ALEXA";
    doc["language"] = "en-US";
    char jsonBuffer[512];
    serializeJson(doc, jsonBuffer);
    client.publish(AWS_IOT_PUBLISH_TOPIC, jsonBuffer);
}
```

## Criação de uma skill para a Amazon Alexa

A Amazon Alexa fornece alguns tipos de serviços como exemplo a serem alterados: Smart Home Skills que são modelos já existentes para o uso de conexão com aparelhos na nuvem, Game Skills com jogo interativo, Music & Audio Skills para músicas e controlar a playlist, Food Ordering Skill que procura restaurantes próximos ou que pode realizar pedidos via comando de voz, Video Skills para visualizar e controlar vídeos por comando de voz, Flash Briefing & News Skills que apresenta as principais notícias baseado no perfil do usuário, Connected Vehicle Skills para controlar e verificar o status de seus veículos conectados de sua casa ou em movimento e Custom Skills as quais são habilidades customizáveis para que o desenvolvedor criar conforme sua necessidade. Para mais informações referentes a exemplos de Skills é possível visualizar no site:

<https://developer.amazon.com/en-US/alexa/alexa-skills-kit/get-deeper>

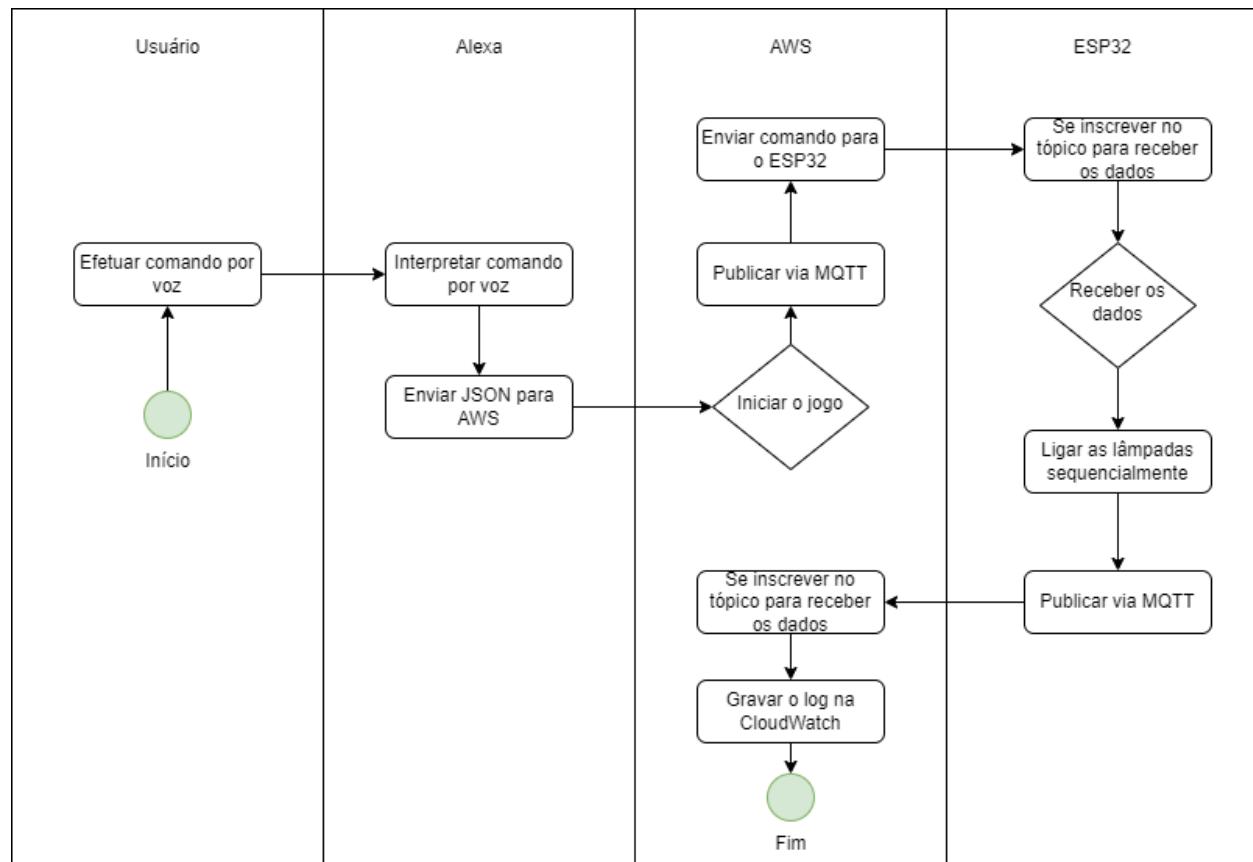
É interessante para quem tem o domínio, ou entende, o idioma inglês verifique o canal oficial dos desenvolvedores da Alexa “Alexa Developers”. Segue link com lista de reprodução com uma série de vídeos com desenvolvimentos e conceitos da Alexa de forma básica:

[https://www.youtube.com/playlist?list=PLdZn93YfA\\_1ZP1WFkz6bm08v3zFfWwfGW](https://www.youtube.com/playlist?list=PLdZn93YfA_1ZP1WFkz6bm08v3zFfWwfGW)

## Diagrama de Atividades

O processo inicia-se pelo comando de voz efetuado pelo usuário. A Amazon Alexa sintetiza o comando em formato JSON para o servidor AWS, que em seguida publica via MQTT. O ESP32 inscreve-se no tópico para receber os dados e, ao receber, liga as lâmpadas sequencialmente. Por fim, o ESP32 publica via MQTT para o AWS, que armazena o log na CloudWatch.

Figura 7 - Diagrama de Atividades



Conforme documentação oficial da Amazon o fluxo de trabalho a seguir demonstra como uma habilidade do Alexa funciona. Neste exemplo, o usuário invoca uma habilidade simples do Alexa chamada "Hello World".

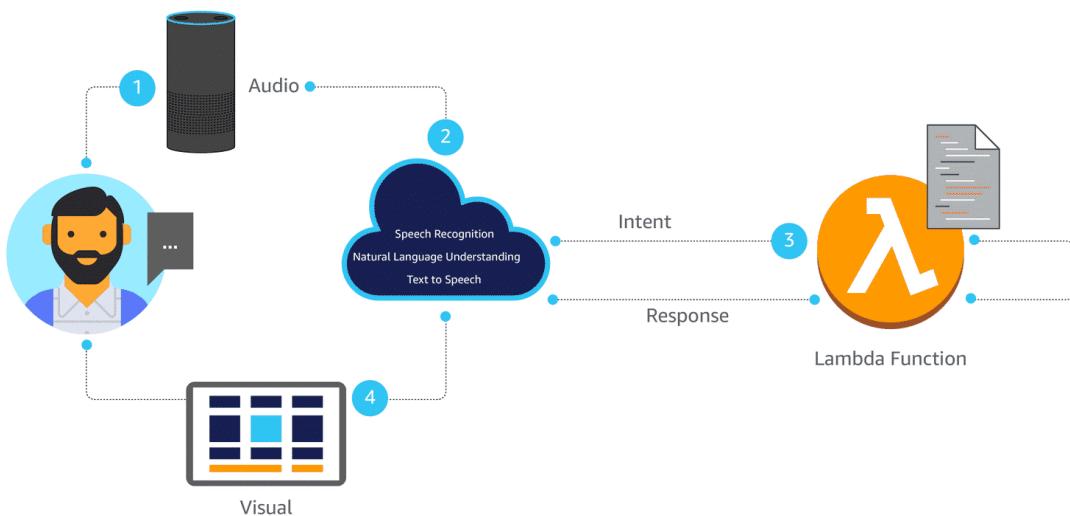
Passo 1 - Para iniciar a habilidade, o usuário diz: "Alexa, abra o Hello World". Uma declaração ou solicitação completa ao Alexa pelo usuário é chamada de enunciado.

Passo 2 - O dispositivo habilitado para Alexa envia o enunciado para o serviço Alexa na nuvem. O serviço Alexa processa o enunciado por meio do reconhecimento automático de fala para converter o enunciado em texto e, em seguida, por meio do entendimento de linguagem natural para reconhecer a intenção do texto.

Passo 3 - O Alexa envia uma solicitação JavaScript Object Notation (JSON) para manipular a intenção para o serviço de computação de recursos que hospeda sua habilidade (geralmente uma função do AWS Lambda). O serviço de computação de recursos atua como back-end e executa seu código de habilidade para lidar com a intenção. Nesse caso, o servidor retorna: "Bem-vindo à habilidade Hello World".

O diagrama na Figura 8 demonstra o que acontece quando um usuário interage com uma habilidade do Alexa. Ele pressupõe que você esteja usando o serviço de computação sem servidor do AWS Lambda para hospedar seu código de habilidade.

Figura 8 - Diagrama da skill Hello World



Esta documentação você pode encontrar no link: <https://developer.amazon.com/en-US/docs/alexa/workshops/build-an-engaging-skill/why-build/index.html#how-an-alexa-skill-works>

E o vídeo com esta explicação se encontra no link a seguir:  
<https://www.youtube.com/watch?v=cakcsuzS2DY&t=17s>

## Conceitos

Neste item será apresentado como realizar a criação de uma skill customizada para a Alexa utilizando os recursos do console do desenvolvedor da Alexa, AWS Lambda, DynamoDB, CloudWatch e IoTCore. Mas antes se faz necessário entender alguns conceitos importantes: Alexa ajuda com várias etapas no processamento de entrada. Primeiro, ela pega a fala do usuário e a transforma em texto escrito (fala em texto). Posteriormente, ele usa um modelo de linguagem para entender o que o usuário quer dizer (compreensão da linguagem natural).

Existem duas formas de prover o servidor, deixar hospedado na própria Alexa ou vincular a um terceiro que neste caso será o AWS Lambda.

Um modelo de interação simples para o Alexa consiste nos seguintes elementos: Nome de invocação, intenções, enunciados, slots.

### Nome de invocação

Um 'nome de invocação' é a palavra ou frase usada para acionar sua habilidade. De certa forma, é o equivalente da voz a um ícone de aplicativo. Esse nome de invocação geralmente corresponde ao nome da sua habilidade, mas dadas as regras sobre a escolha de um nome de invocação, poderá ser sendo um pouco diferente.

## Intenções

Uma intenção, ou *intent*, é o que um usuário está tentando realizar. Dentro do código, é o método a ser executado. Intenção não se relaciona com as palavras específicas que um usuário diz, mas com o objetivo de alto nível que ele almeja.

## Declarações

Declarações são as frases específicas que os usuários usarão ao fazer uma solicitação após a abertura da skill e que por consequência executará uma determinada ‘Intenção’. Muitas vezes, há uma grande variedade de enunciados que se encaixam na mesma intenção. E às vezes pode até ser um pouco mais variável.

## Slots

Um slot é uma variável relacionada a uma intenção que permite ao Alexa entender as informações sobre a solicitação. Por exemplo, em uma skill que sugere um filme aleatório o usuário poderá falar qual gênero do filme ele quer assistir, sendo assim o gênero é a variável que o sistema precisa interpretar de forma não fixa.

## Tipos de slots

A Amazon fornece vários tipos de slots integrados, como datas, números, durações, hora etc. Mas os desenvolvedores podem criar slots personalizados para variáveis específicas de sua skill.

## Console do desenvolvedor Alexa

A partir da Figura 9 até Figura 23 é apresentado como configurar uma nova Skill. Será utilizado a localização do Brasil, ressalto que existem algumas funcionalidades disponíveis apenas em outras localidades então o layout da tela poderá ser diferente dependendo do idioma.

Figura 9- Página inicial do site da Amazon Alexa

The screenshot shows the Alexa Skills Kit landing page. At the top, there's a navigation bar with links for 'Skill Builders', 'Device Makers', 'Produtos', 'Programas', 'Docs', and 'Blog'. A search bar and a 'Sign In' button are also present. The main content area features a large teal banner with the title 'Curso de Zero a Herói' and a description about a course. Below the banner, there are several buttons: 'Alexa Skills Kit' (highlighted with a red box), 'Comece Agora', 'Aprofunde-se', 'Cresça seu Negócio', 'Novidades', and 'Console' (also highlighted with a red box). A breadcrumb navigation 'Alexa > Alexa Skills Kit' is visible. The bottom section contains a heading 'O que são Alexa Skills?' with a detailed description of what skills are and how they work. To the right of the text is an image showing various Alexa devices like Echo, Fire TV, and a tablet displaying a skill interface.

Figura 10 - Página de login para o console do desenvolvedor

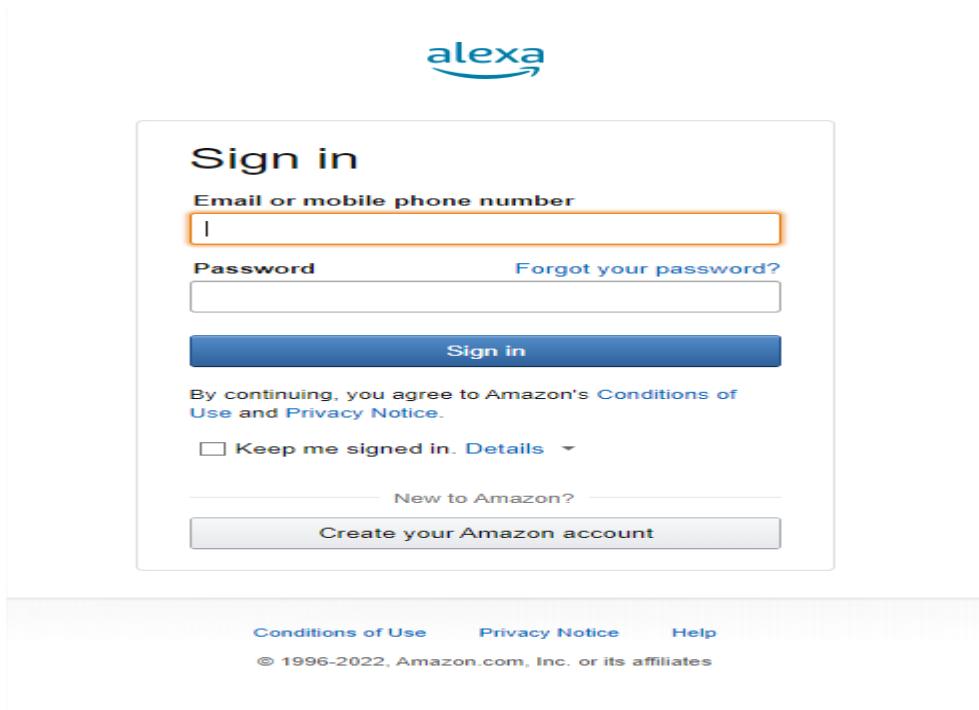


Figura 11 - Página para gerenciamos das skills

The screenshot shows the 'Skills' management page in the Alexa developer console. At the top left is the Alexa developer console logo. The top navigation bar includes links for 'Skills', 'Earnings', 'Payments', 'Hosting', and 'Settings'. A search bar with the placeholder 'Search Alexa Developer help...' and a magnifying glass icon is located at the top right. A success message 'livro\_miguel has successfully been deleted.' is displayed with a checkmark icon. The main content area is titled 'Alexa Skills' and shows a table of skills. The table columns are 'SKILL NAME', 'LANGUAGE', 'MODIFIED', 'STATUS', and 'ACTIONS'. The skills listed are: 'TCC V1N' (Portuguese (BR) +1, 2022-09-25, In Dev), 'TEST\_ESP32' (English (US), 2022-09-21, In Dev), 'TCC V1' (Portuguese (BR) +1, 2022-08-28, In Dev), 'Raízes de uma equação de 2 grau' (Portuguese (BR), 2022-08-13, In Dev), and 'Sistema de Cadastro' (Portuguese (BR), 2022-08-13, In Dev). Each skill row has a 'Choose action' dropdown menu. To the right of the table is a large 'Create Skill' button highlighted with a red box. On the right side of the page, there is a 'alexा LIVE' section with a video thumbnail, a 'Alexa Skill Insights' section with a cartoon character icon, and a 'To Dos' section. The bottom of the page includes a language selector 'English (US)', a 'Feedback' link, and a footer with copyright information: '© 1996-2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms | Docs | Forums | Blog | Alexa Developer Home'.

Figura 12 - Selecionar o idioma e modelo Custom

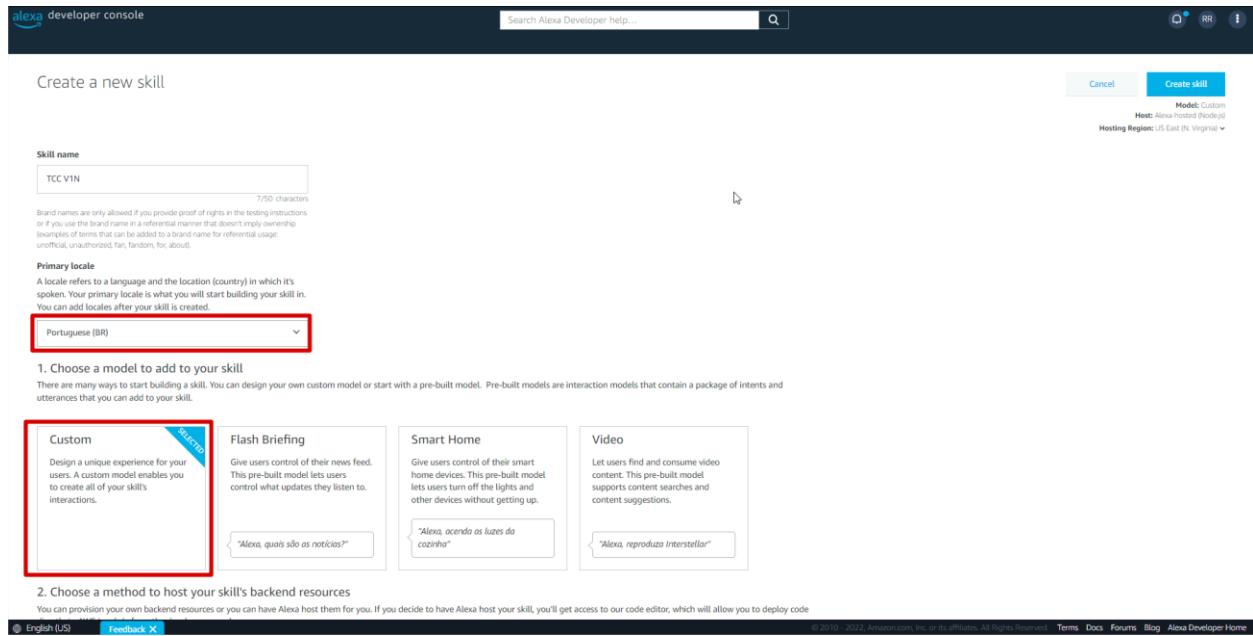


Figura 13 – Selecionar a linguagem de programação

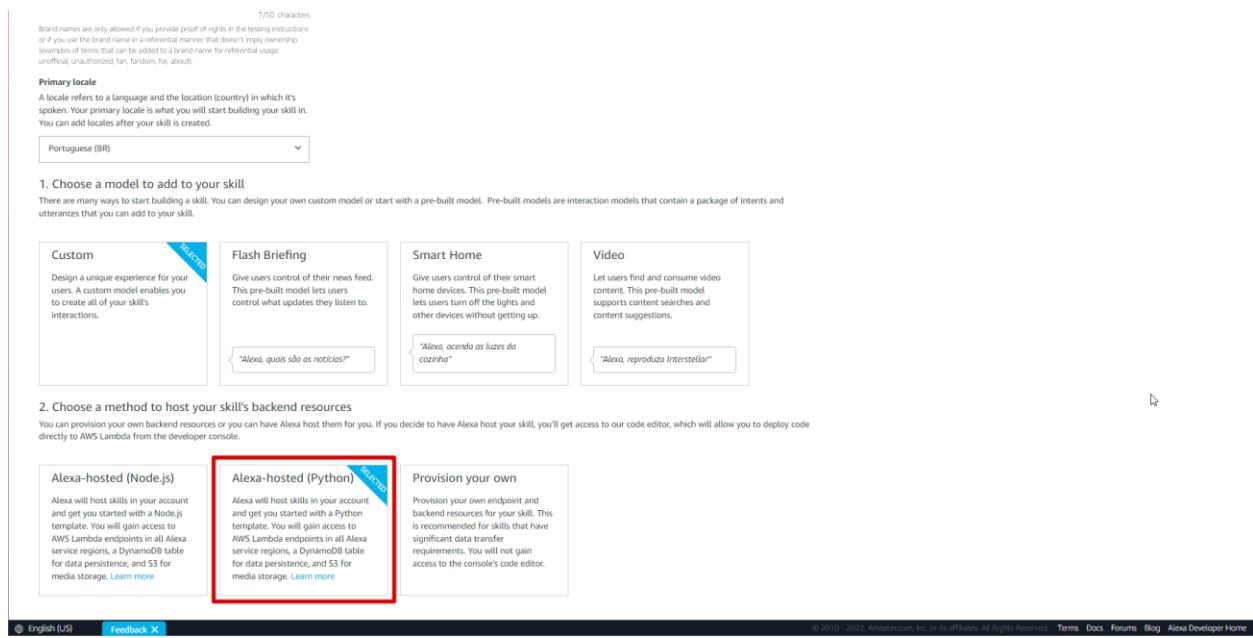


Figura 14 - Selecionar “Start from Scratch”

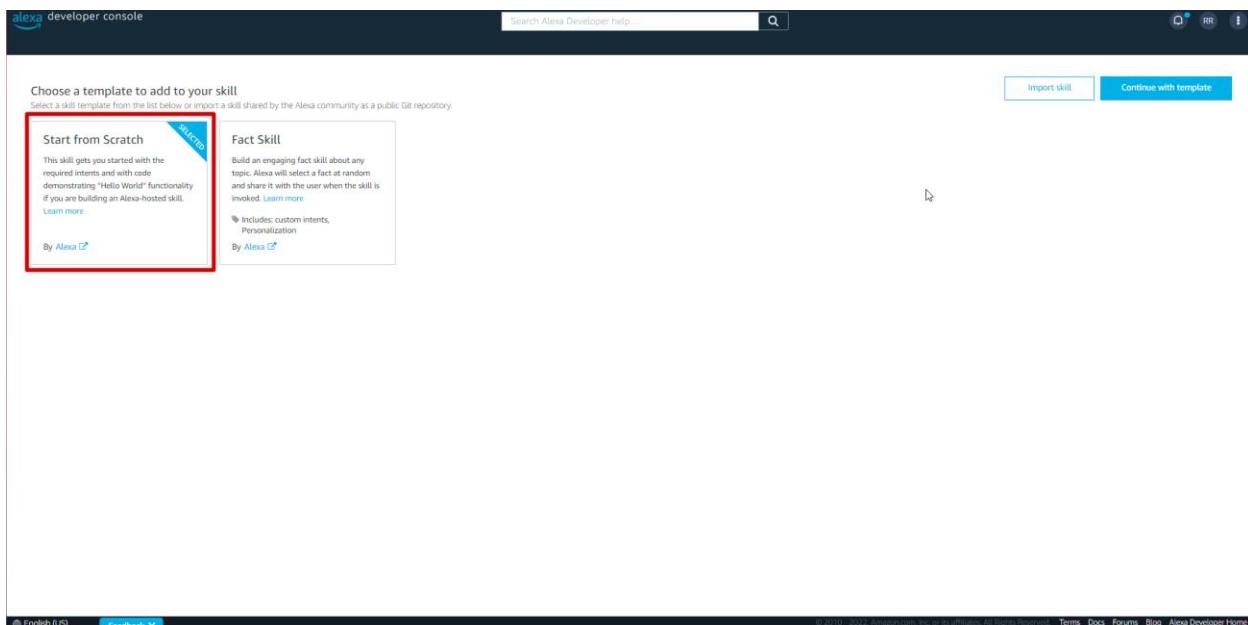


Figura 15 - Inserindo um nome de invocação da skill

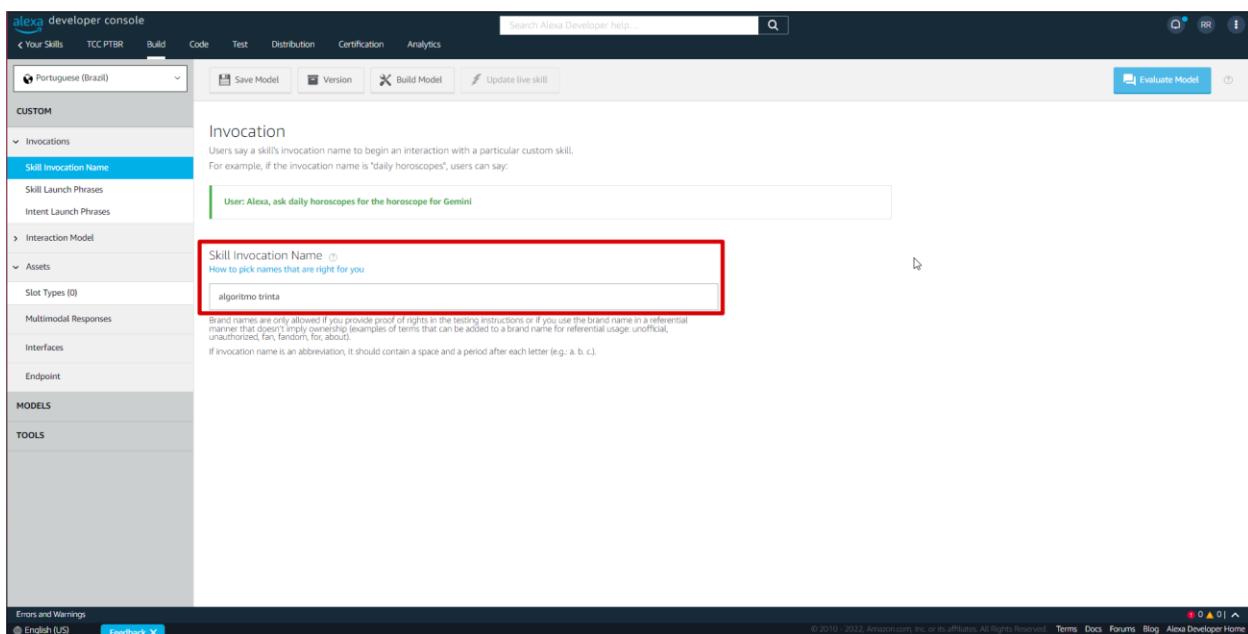


Figura 16 - Adicionando novo intent

The screenshot shows the Alexa developer console interface. On the left, there's a sidebar with various skill components like Skill Invocation Name, Skill Launch Phrases, Intent Launch Phrases, and an Interaction Model section expanded to show Intents (11). The 'Intents' section on the right lists several intents with columns for NAME, UTTERANCES, SLOTS, TYPE, and ACTIONS. A prominent red box highlights the blue '+ Add Intent' button at the top of the intent list.

Figura 17 - Criando um intent personalizável

This screenshot shows the detailed configuration for the 'ShowRankingIntent'. The left sidebar has 'ShowRankingIntent' selected. The main area shows a sample utterance 'Fale a minha pontuação jogador [playerName] [playerLastName]'. Below it, the 'Dialog Delegation Strategy' section notes that dialog management is disabled. The 'Intent Slots' section lists a single slot named 'playerName' with type 'AMAZON.FirstName'. The bottom status bar indicates 0 errors and 0 warnings.

Figura 18 - Criando um intent e slot personalizável

The screenshot shows the Alexa developer console interface for creating a skill. On the left, a sidebar lists various intents and built-in intents. The 'ScoreIntent' intent is currently selected. The main area displays the 'Intents / ScoreIntent' configuration. Under 'Sample Utterances', there are three examples: '(SequenceColors)', '(SequenceColors)', and '(SequenceColors)'. Below this is the 'Dialog Delegation Strategy' section, which is currently disabled. The 'Intent Slots' section shows one slot named 'SequenceColors' with the type 'Colors' highlighted with a red box. A second slot, 'Create a new slot', is listed below it. At the bottom, there are tabs for 'Errors and Warnings' and 'Feedback'.

Figura 19 - Adicionando valores no slot personalizado

The screenshot shows the 'Slot Types / Colors' page in the Alexa developer console. The 'Colors' slot type is selected. The 'Slot Values' section lists four entries: 'amarelo', 'azul', 'verde', and 'vermelho'. Each entry has fields for 'VALUE', 'ID (OPTIONAL)', and 'SYNONYMS (OPTIONAL)'. The 'vermelho' entry is highlighted with a red box. Below this is the 'Slots Using (Colors)' section, which is currently empty. The bottom of the screen shows navigation tabs like 'Your Skills', 'TCC PTBR', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics', along with a search bar and footer links.

No item apresentado na Figura 20 você deverá cadastrar qual declaração para ativar essa intenção.

Figura 20 - Inserindo uma declaração

The screenshot shows the Alexa developer console interface. The left sidebar lists various intent categories under 'CUSTOM'. The 'Intents (11)' section is expanded, showing 'InitializeGameIntent' with two slots: 'difficulty' and 'gametype'. The 'Sample Utterances (4)' section contains four entries: 'iniciar o jogo no modo [difficulty] e [gametype]' (highlighted with a red box), '[gametype]', '(difficulty)', and '(difficulty) [gametype]'. Below this is the 'Dialog Delegation Strategy' and 'Intent Slots (2)' sections.

Neste exemplo essa declaração serve apenas para iniciar o jogo escolhendo o modo e a dificuldade com dois slots personalizados.

Após realizado todas as alterações deve-se salvar e compilar o modelo.

Figura 21 - Salvar e construir o modelo

This screenshot is similar to Figura 20, showing the Alexa developer console. The 'Build' tab is now active. The 'Save Model' and 'Build Model' buttons are highlighted with red boxes. The rest of the interface, including the 'Intent Utterances' section with the highlighted utterance, remains the same.

Figura 22 - Mensagem de sucesso após construção do modelo



Também está disponibilizado no github o JSON que pode ser importado para realizar todos esses cadastrados de forma automática.

Figura 23 - Importando JSON

A screenshot of the Alexa Developer Console showing the "JSON Editor" section. On the left, there's a sidebar with various categories like "Your Skills", "TCC PTBR", "Build", "Code", "Test", "Distribution", "Certification", and "Analytics". Under "Build", "Intent History" is selected. The main area shows a "JSON Editor" with a "Save Model", "Version", "Build Model", and "Evaluate Model" button bar. Below that is a search bar and a "Drag and drop a .json file" placeholder. The central part contains a large JSON code block with line numbers from 1 to 45. The JSON defines interaction models, language models, intents, slot types, and game types. At the bottom, there are "Errors and Warnings" and "Feedback" buttons.

Após isso, toda vez que é executada uma intenção será gerado um Callback para o servidor Lambda executando o método vinculado a esta intenção.

## Criando servidor AWS Lambda

Nesta etapa, deverá ser criada uma função no AWS Lambda. O site para utilizar esses serviços seria o: <https://aws.amazon.com>

Os passos são descritos a partir de Figura 24 até a Figura 41.

Figura 24 - Página inicial da AWS Amazon

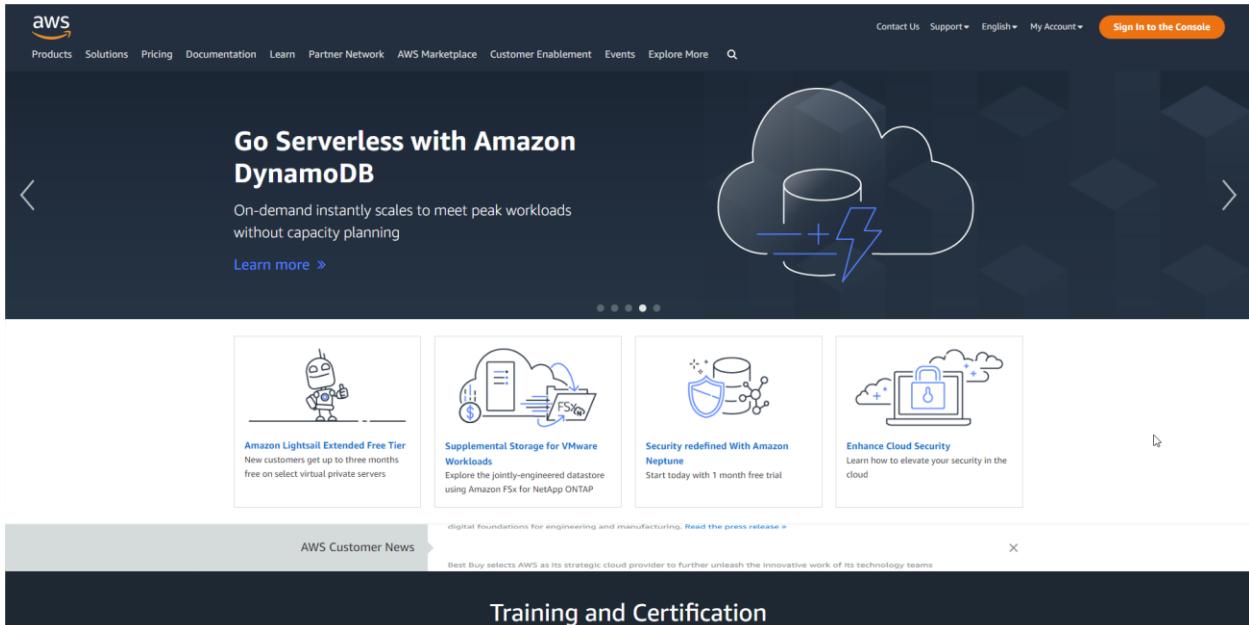


Figura 25 - Página de login da AWS Amazon

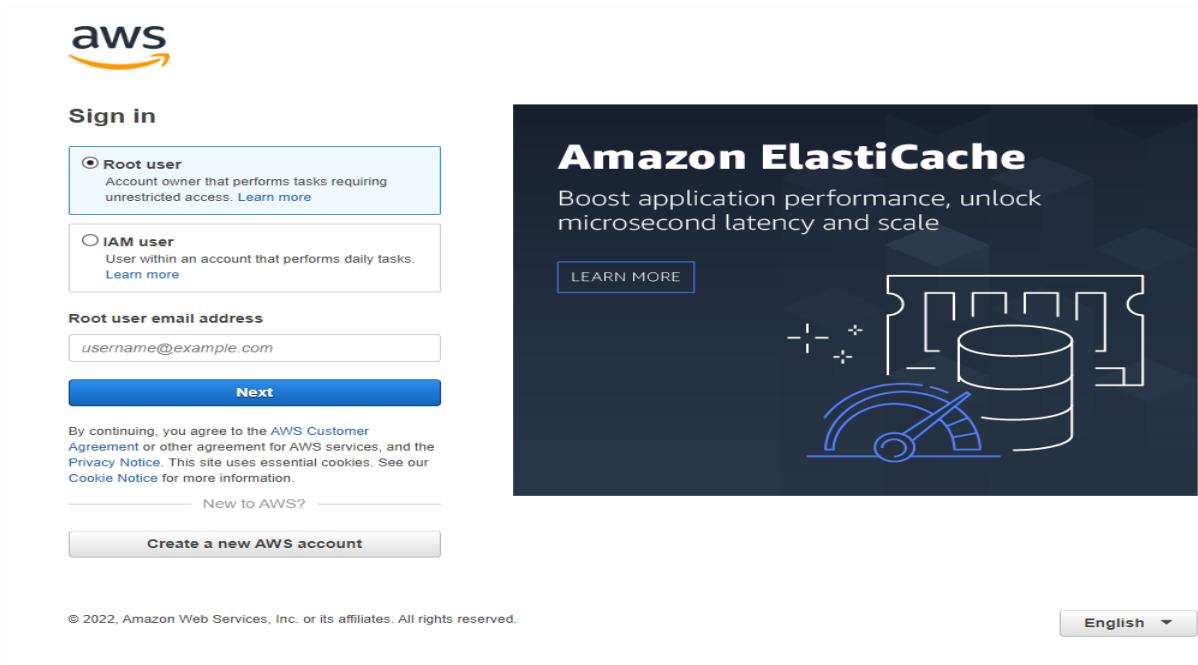


Figura 26 - Selecionar o serviço Lambda

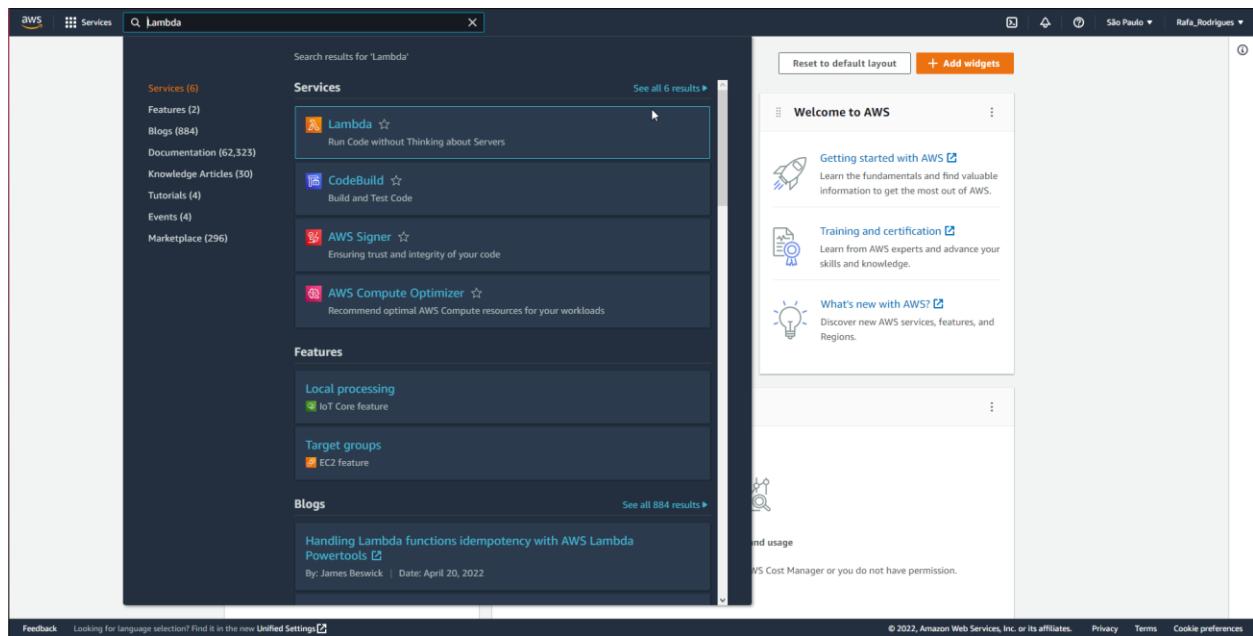


Figura 27 - Criar a função

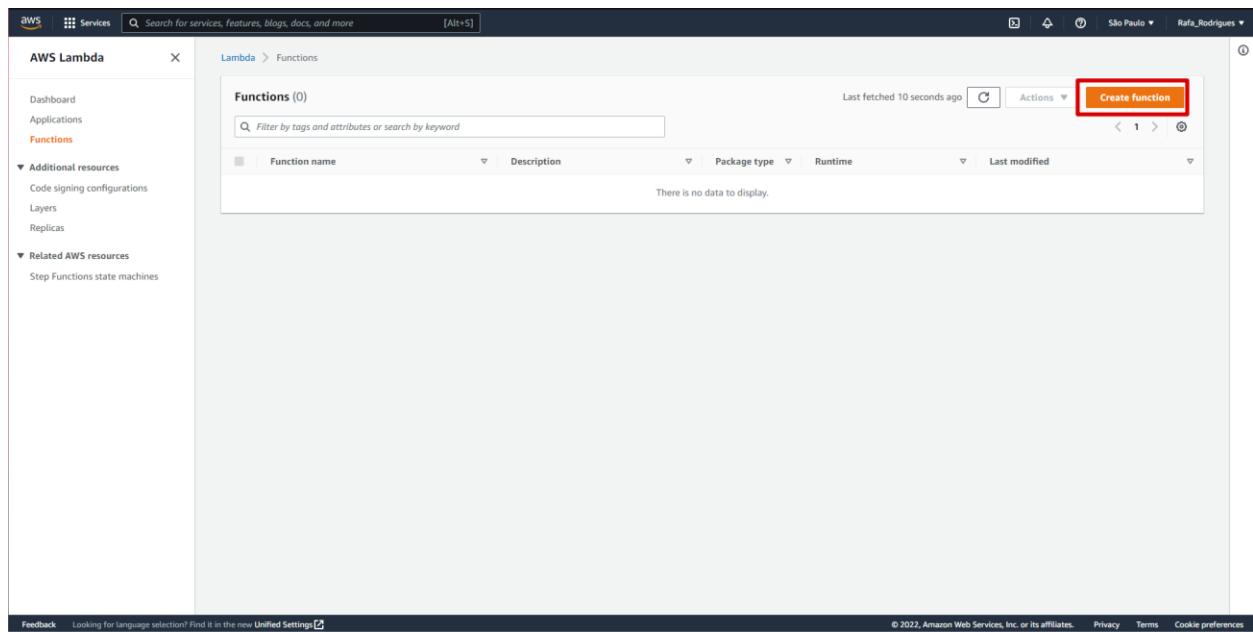


Figura 28 - Escolher um nome e linguagem de programação

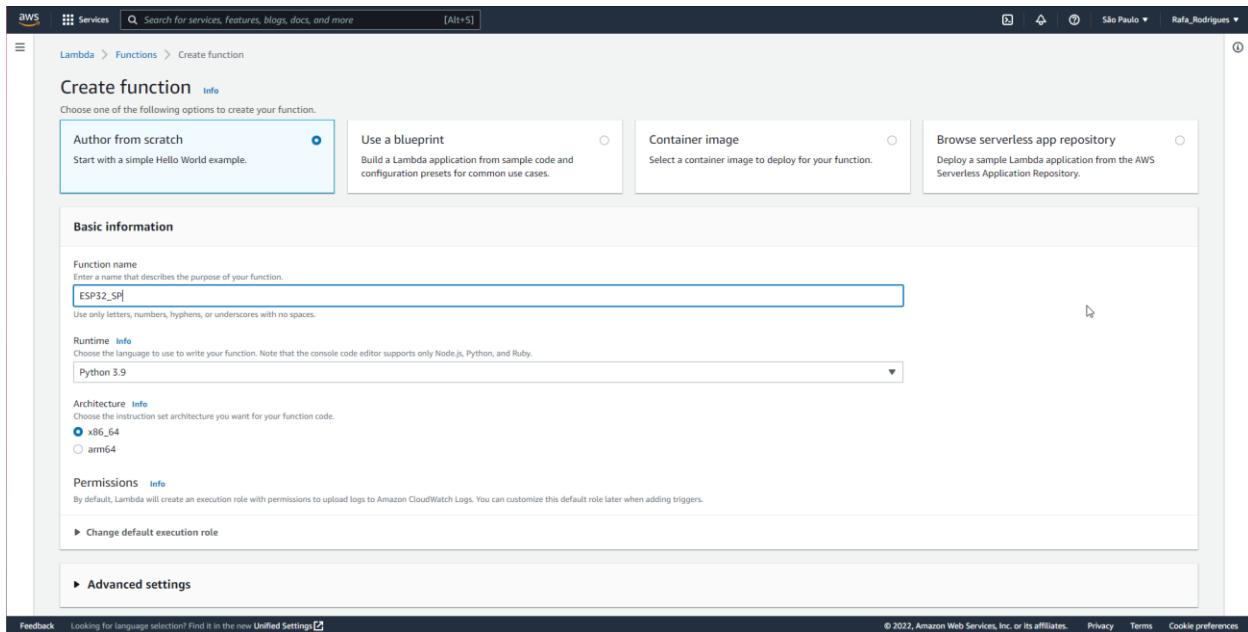
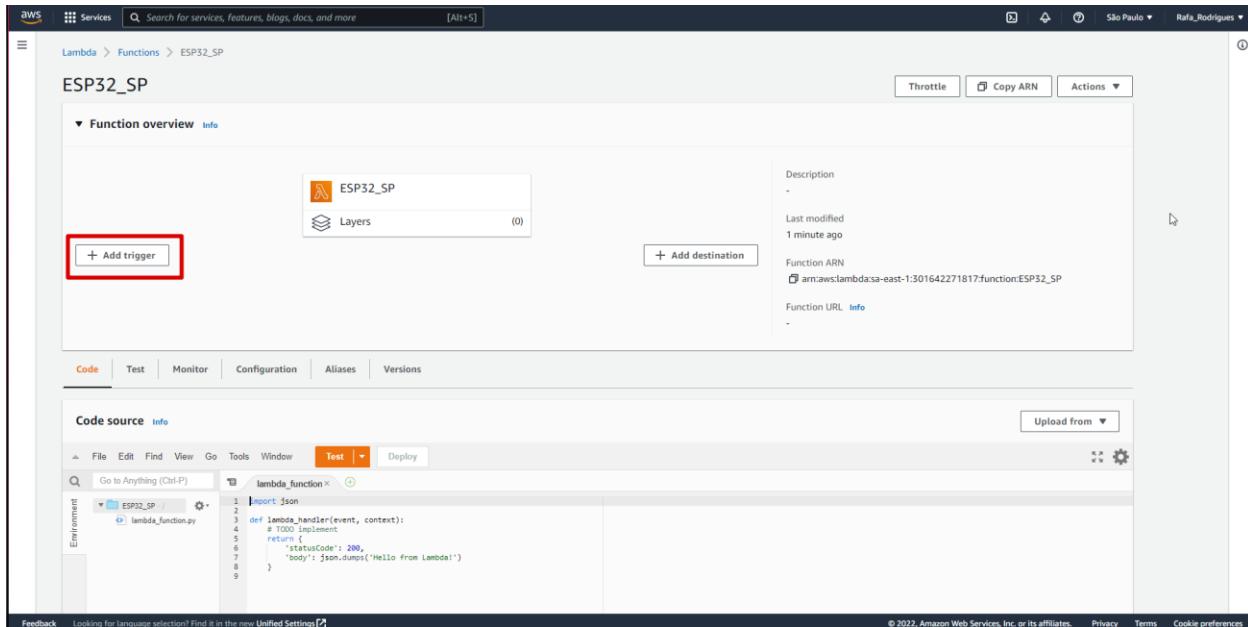


Figura 29 - Adicionar um gatilho



Nesta etapa será necessário o ID gerado na Amazon Alexa. A Figura 30 aponta o lugar que se obtém na Amazon Alexa. Na Figura 31 mostra onde deve ser inserido o ID da Amazon Alexa. Após a inserção do ID, clicar em salvar.

Figura 30 - Utilizar a opção “Copy to clipboard”

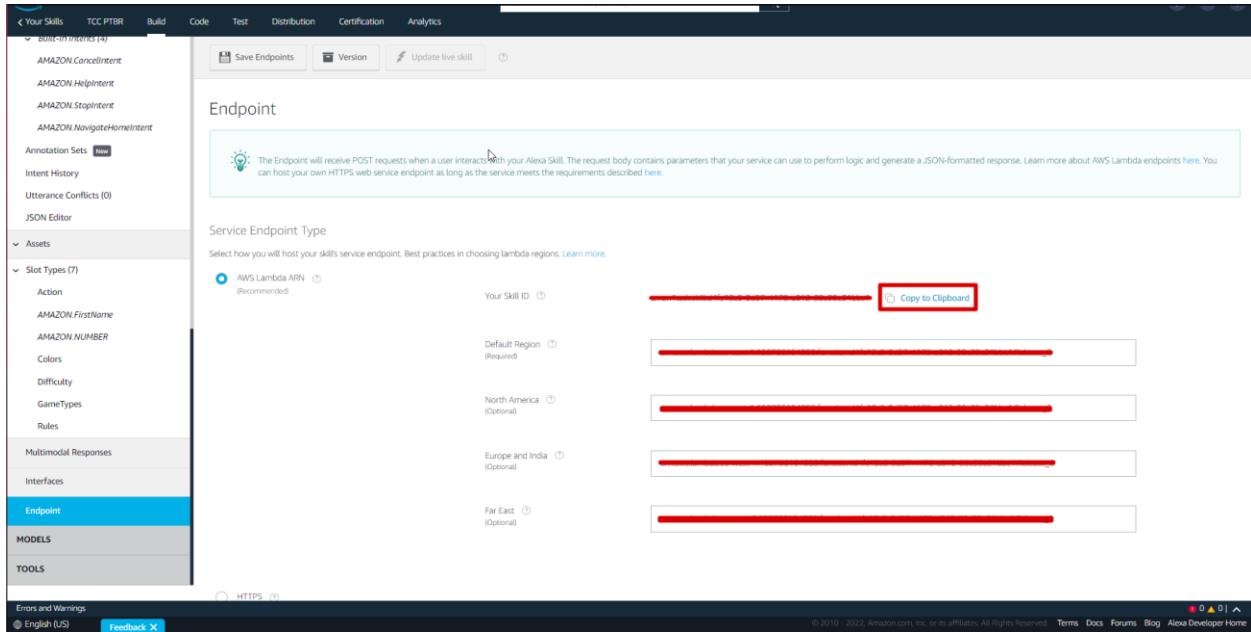
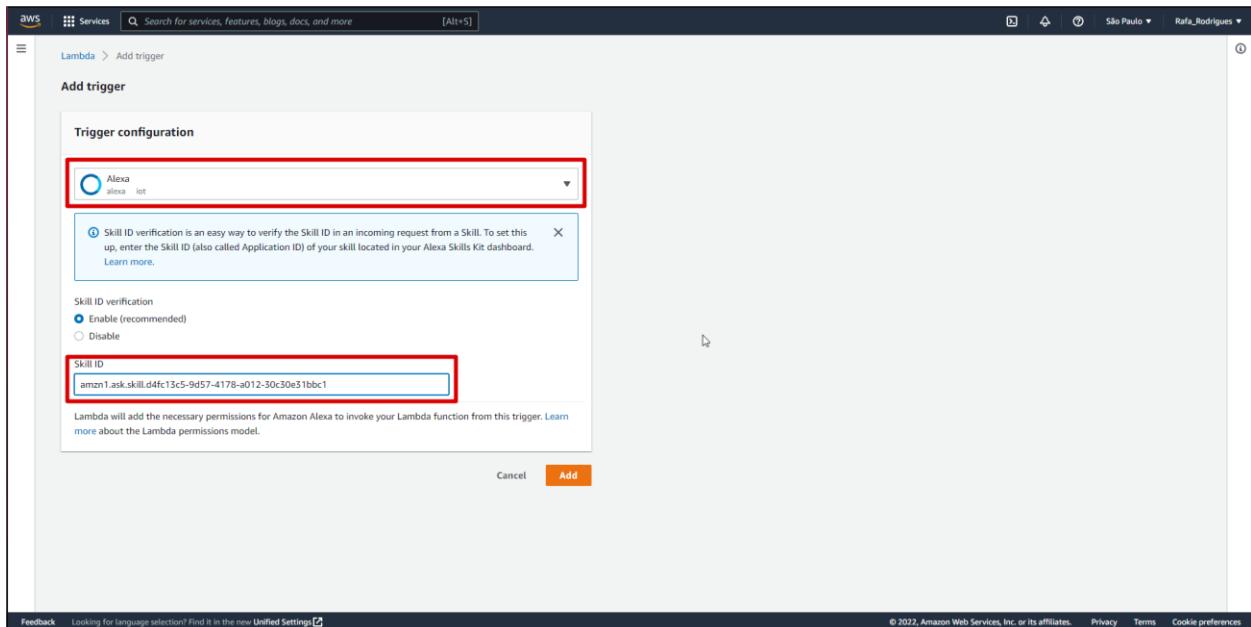


Figura 31 - Escolher “Alexa IoT” e colar o código copiado anteriormente no “Skill ID”



Será necessário realizar o processo inverso, copiar o código ARN da função Lambda e colar no endpoint da Skill para ela conseguir acessar o backend. A Figura 32 aponta onde se obtém na função Lambda. Na Figura 33 mostra onde deve ser inserido o ID na Skill. Após a inserção do ID, clicar em salvar.

Figura 32 - Copiar o “Function ARN”

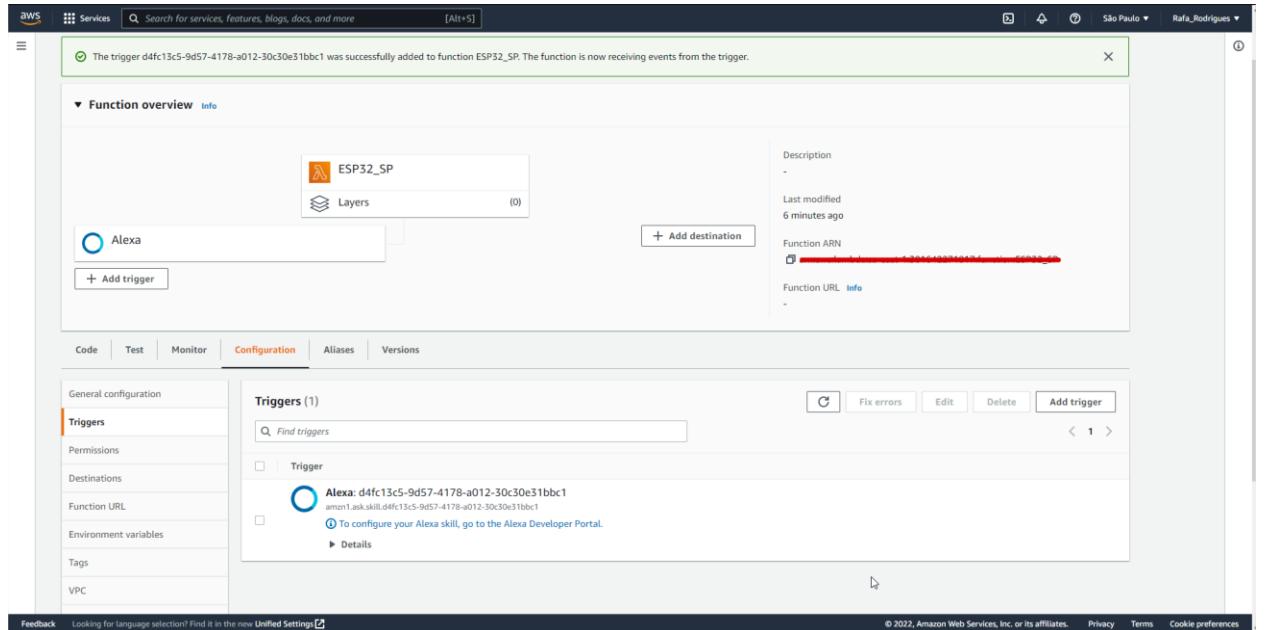
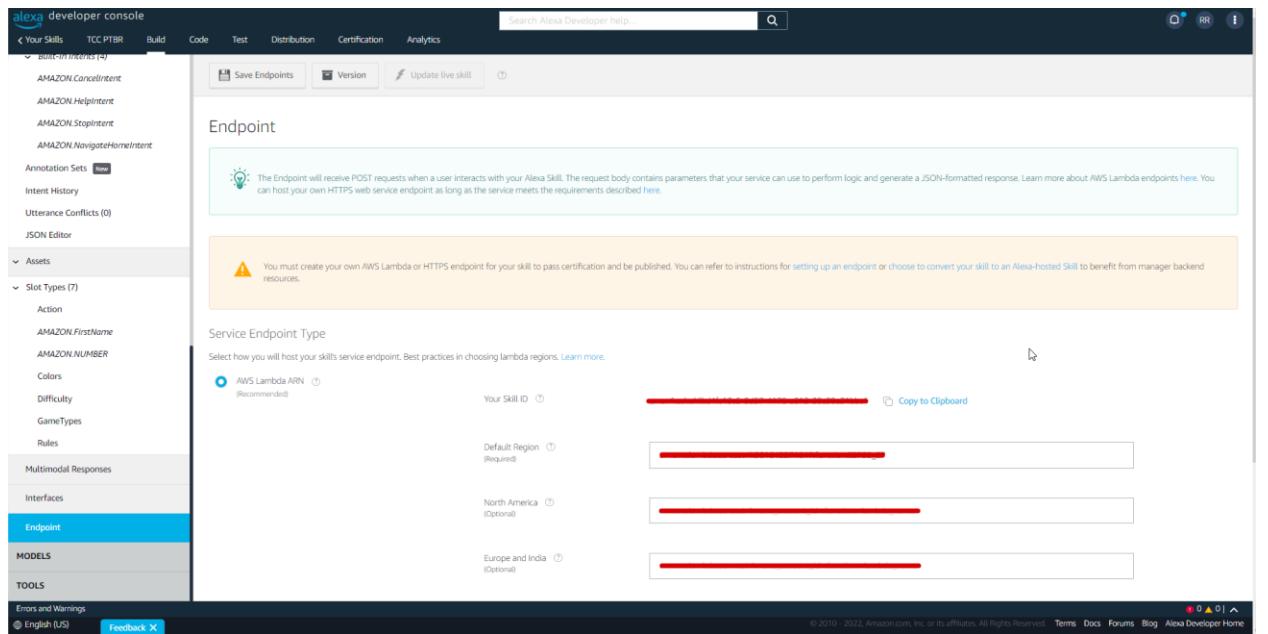


Figura 33 - Colar o “Function ARN”



Deve ser realizado a configuração de permissão para a função criada, para isso deve-se seguir o passo-a-passo da Figura 34 até Figura 37 - Painel atualizadoFigura 37.

Figura 34 - Clicar no hiperlink

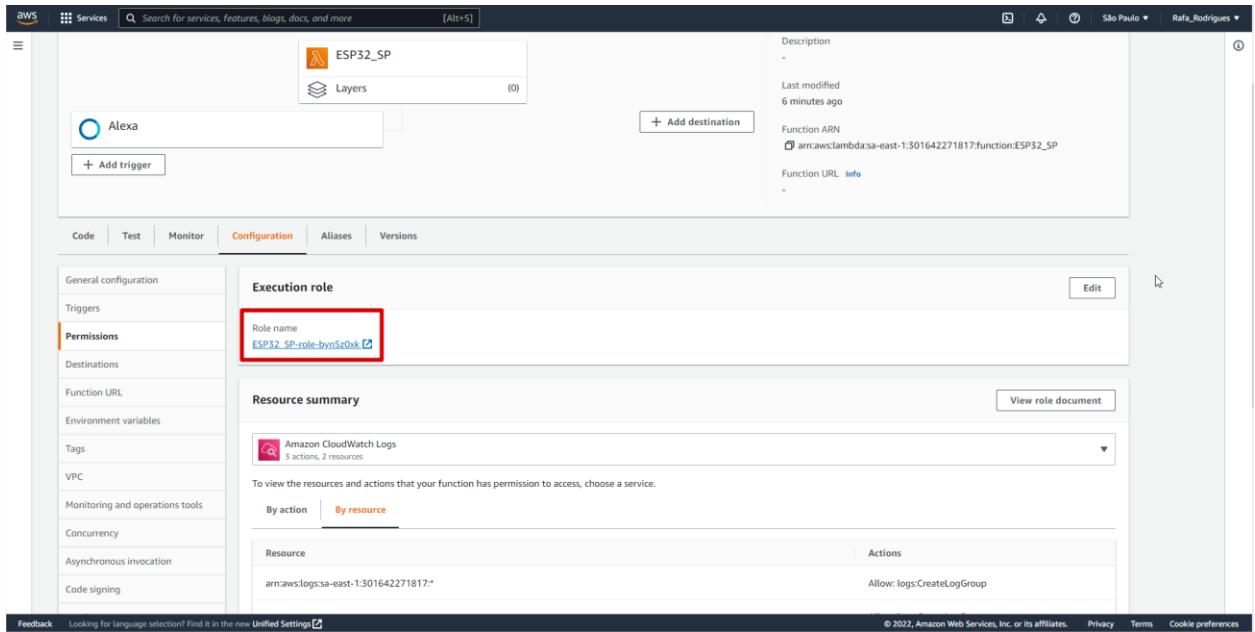
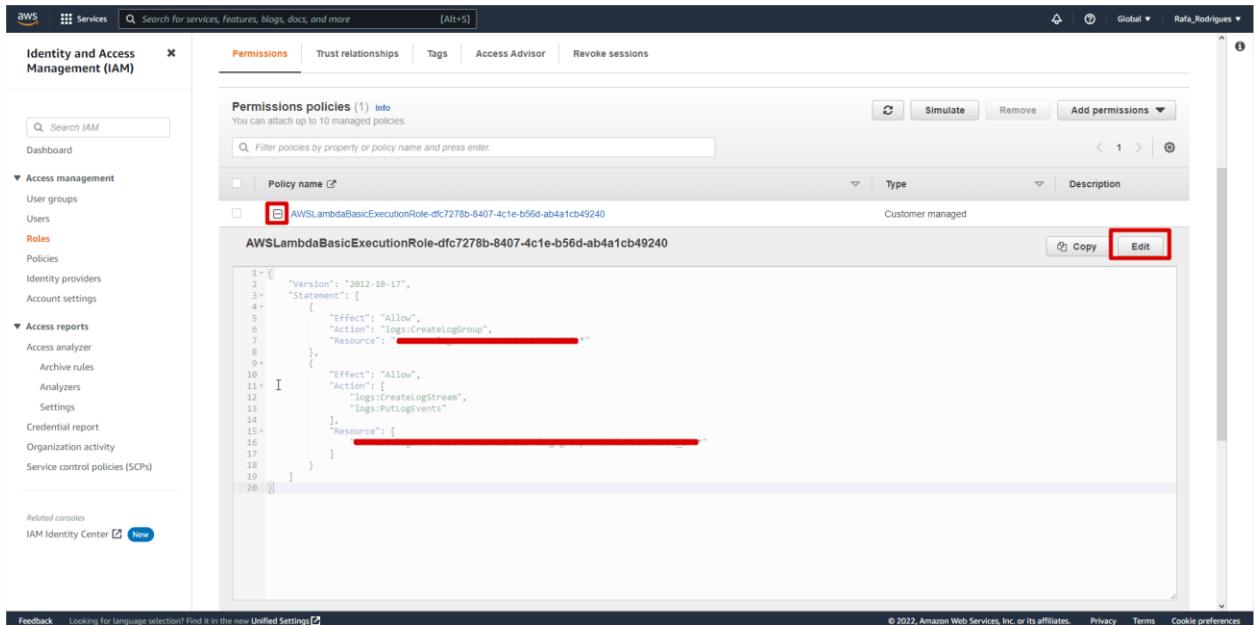
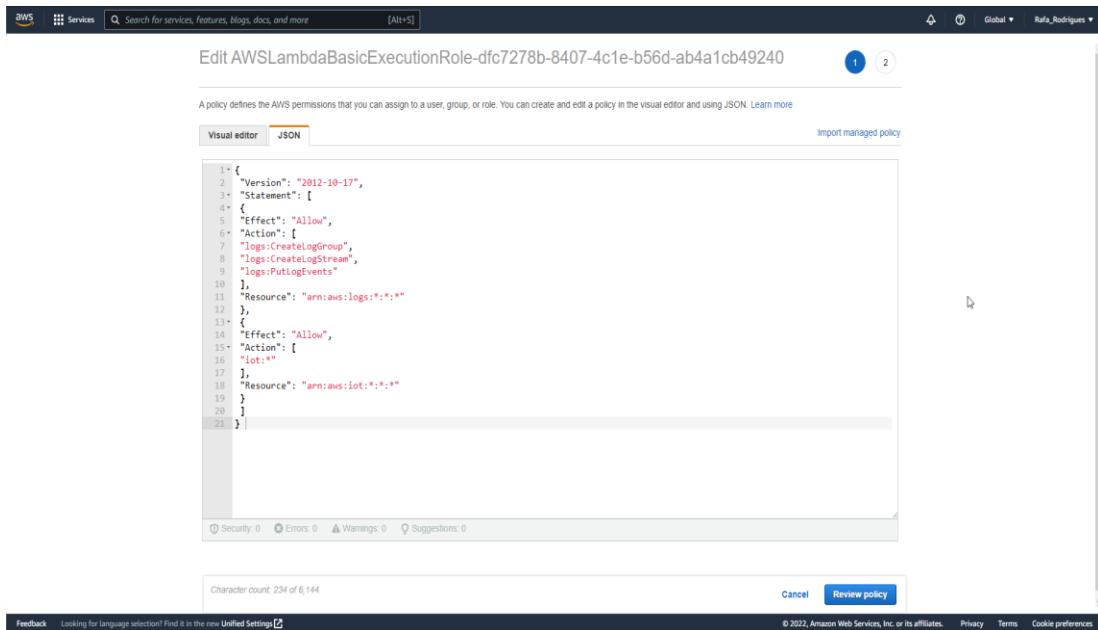


Figura 35 - Expandir e editar a política



Se as permissões corretas não forem concedidas, dificultará o acesso a determinados tópicos ou a execução de determinadas ações. Neste caso, define-se a política no nível mais permissivo. As configurações podem ser alteradas para restringir determinadas ações e tópicos conforme necessários, por isto muita atenção às permissões. Na tela da Figura 36, incluir o código de permissão do AWS IoT seguindo o exemplo do Quadro 1.

Figura 36 - Colar e salvar



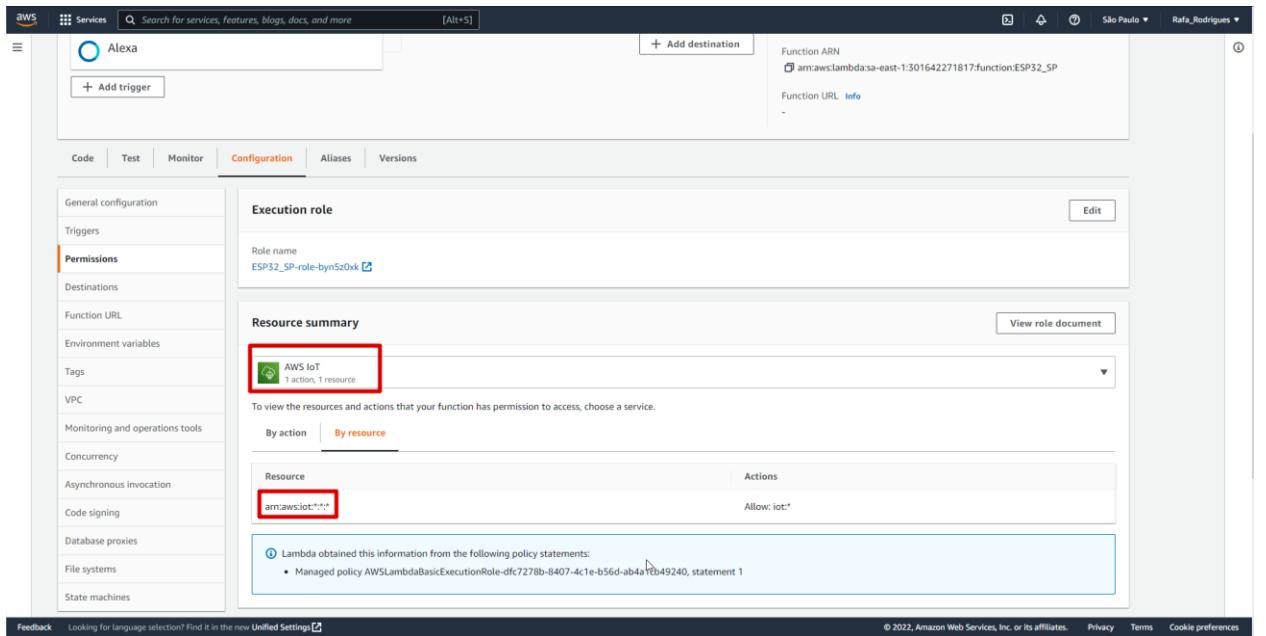
Quadro 1 - Política Utilizada

Código

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:/*"
      ],
      "Resource": "arn:aws:iot:*:*"
    }
  ]
}
```

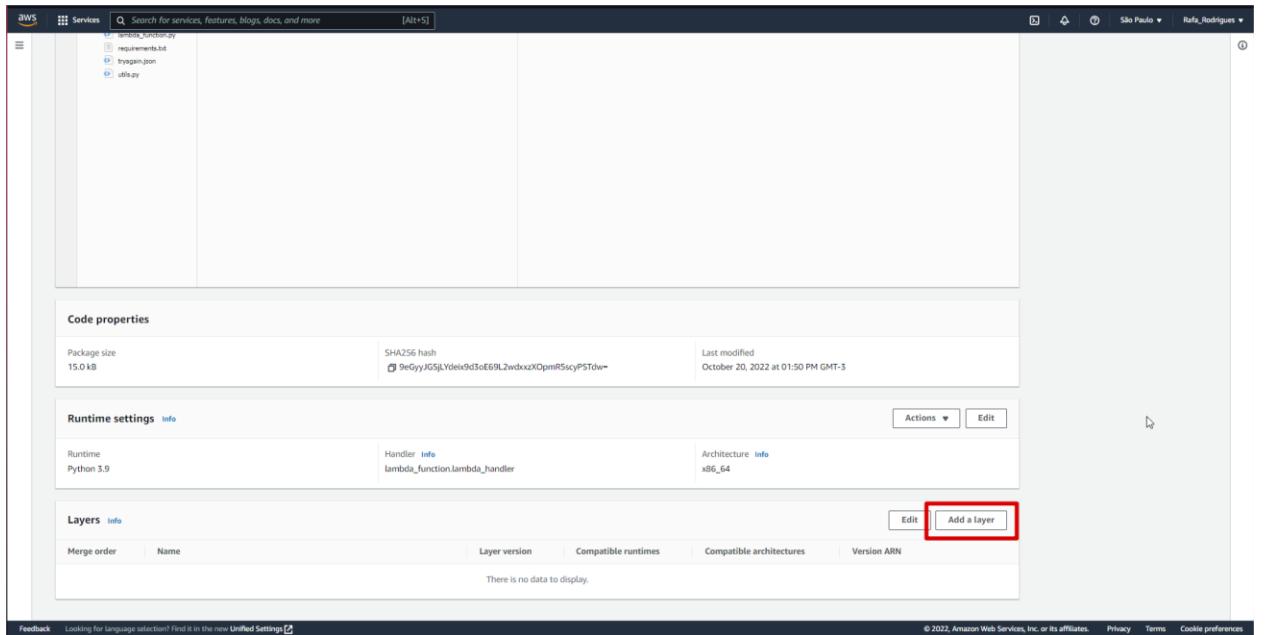
Após salvar e revisar a política a página deverá ficar como Figura 37.

Figura 37 - Painel atualizado



O Lambda não tem acesso as bibliotecas/recursos do pacote ask\_sdk, esse pacote seria o padrão para utilizar os recursos da Alexa com isso é necessário adicionar uma camada para o Lambda ter de onde buscar esse recurso, para isso na parte inferior do site clicar em “Add a Layer”.

Figura 38 - Adicionando uma camada



Após isso, devemos especificar qual camada iremos adicionar no caso já disponibilizaram o pacote ask\_sdk na url “arn:aws:lambda:us-east-1:173334852312:layer:ask-sdk-for-python-36:1”, desta forma deve-se utilizar a opção “Specify na ARN” e colar no campo este valor.

Figura 39 - Especificando a camada

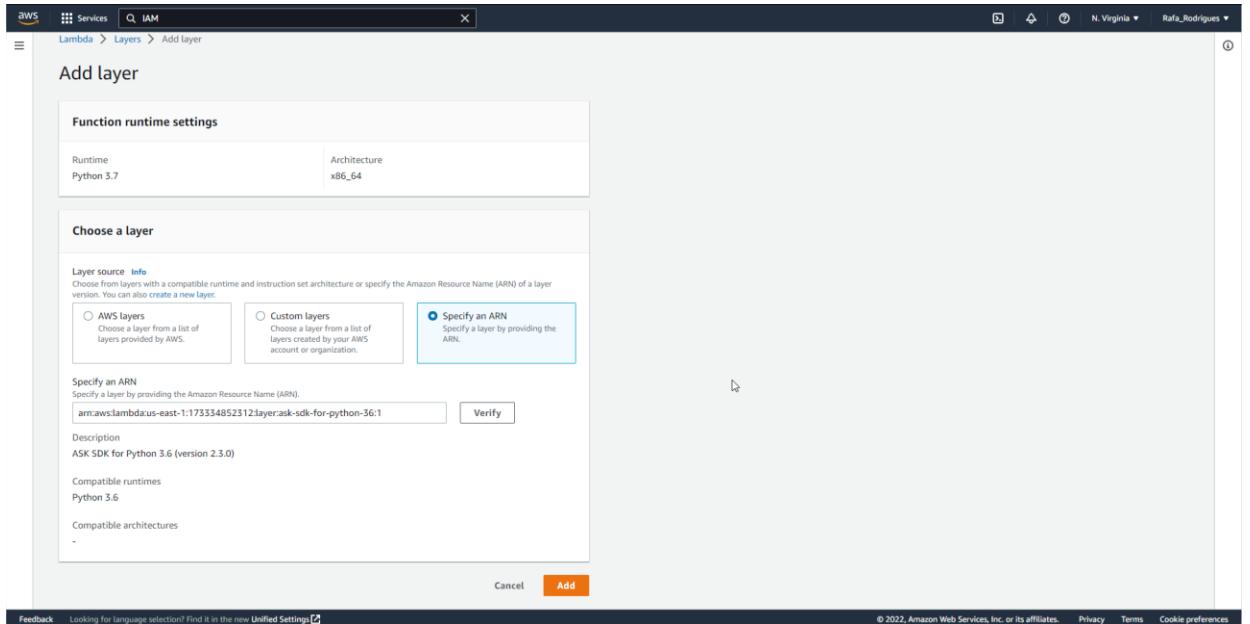
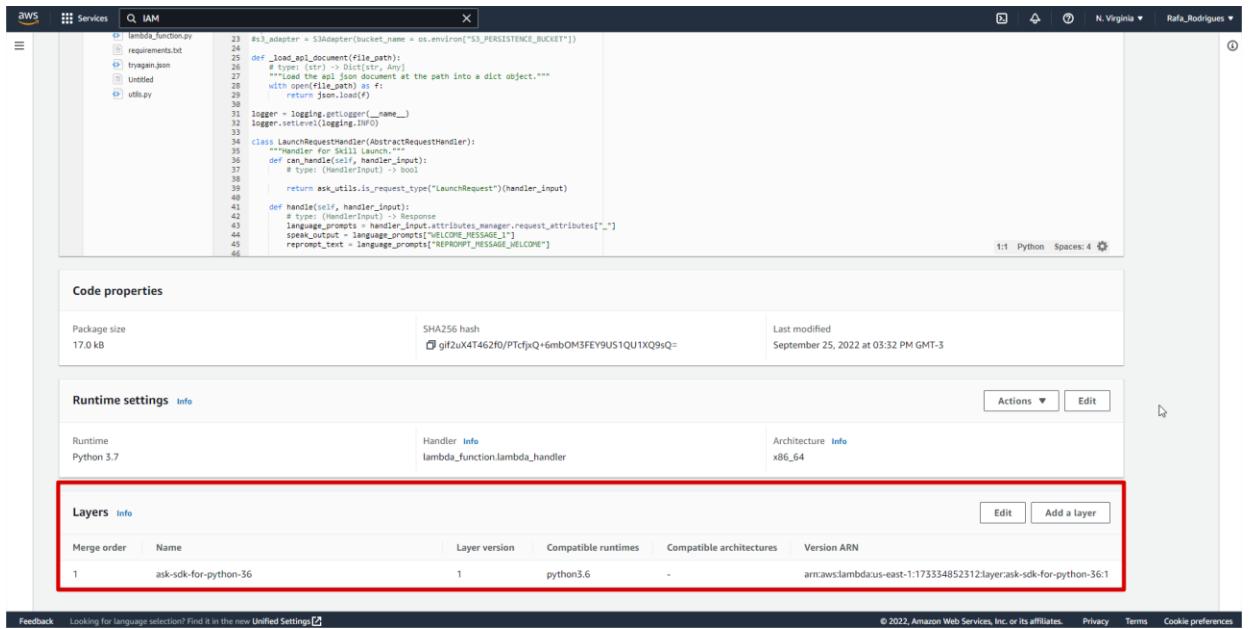


Figura 40 - Como ficará o item camada



No item “Code” é onde deverá ser feito a lógica após receber o JSON da Skill, esse código se encontra no github. Após subir o servidor ele já deverá se comunicar com a skill e vice-versa.

*Figura 41 - Código do backend*

The screenshot shows the AWS Lambda console interface. At the top, there's a search bar and a navigation bar with tabs for Services, Search for services, features, blogs, docs, and more, and [Alt+S]. On the right, there are account and user information: São Paulo and Rafa\_Rodrigues. Below the header, a green banner indicates "Successfully updated the function ESP32\_SP." The main area displays the function configuration for "ESP32\_SP". It includes a thumbnail for the function, a "Layers" section (0 items), a "Description" field (empty), and a "Last modified" timestamp (27 minutes ago). The "Function ARN" is listed as arn:aws:lambda:dasa-east-1:301642271817:function:ESP32\_SP, and the "Function URL" has an "Info" link. A "Code" tab is selected, showing the function's code in a code editor. The code is a Python script named "lambda\_function.py" that handles Alexa skills. The code editor has tabs for Code, Test, Monitor, Configuration, Aliases, and Versions. A "Test" tab is active, and a "Deploy" button is highlighted with a red box. The code editor also has an "Upload from" dropdown and some settings icons. The left sidebar shows the environment and a file tree with "ESP32\_SP" and "lambda\_function.py".

## Alexa Presentation Language

A Alexa tem uma estrutura de design visual chamada Alexa Presentation Language (APL), que permite criar experiências interativas de voz e visual utilizando a tela do dispositivo. APL fornece elementos visuais, incluindo: gráficos, imagens, apresentações de slides e vídeo.

Segundo documentação oficial é possível criar elementos visuais personalizados para dispositivos padrão habilitados para Alexa, como Echo Show, Echo Spot, Fire TV e dispositivos Fire Tablet selecionados. Dispositivos de terceiros criados usando o Alexa Smart Screen e o SDK do dispositivo de TV também suportam a estrutura de design APL.

O APL suporta comandos de voz para que os usuários possam solicitar um item na tela em vez de depender apenas de interações de toque.

A APL cria um arquivo JSON enviado de sua skill que contém as especificações para seus elementos de design. O dispositivo avalia o que pode suportar e, em seguida, importa imagens e outros dados conforme necessário para renderizar a experiência correta.

Exemplos de documentos APL podem ser encontrados no site: <https://apl.ninja>

Neste projeto foram utilizados os JSON disponibilizados nos links: <https://apl.ninja/xeladotbe/you-win-0mfh> e <https://apl.ninja/xeladotbe/jeff-blankenburgs-twitch-stream-intro-k8a9>

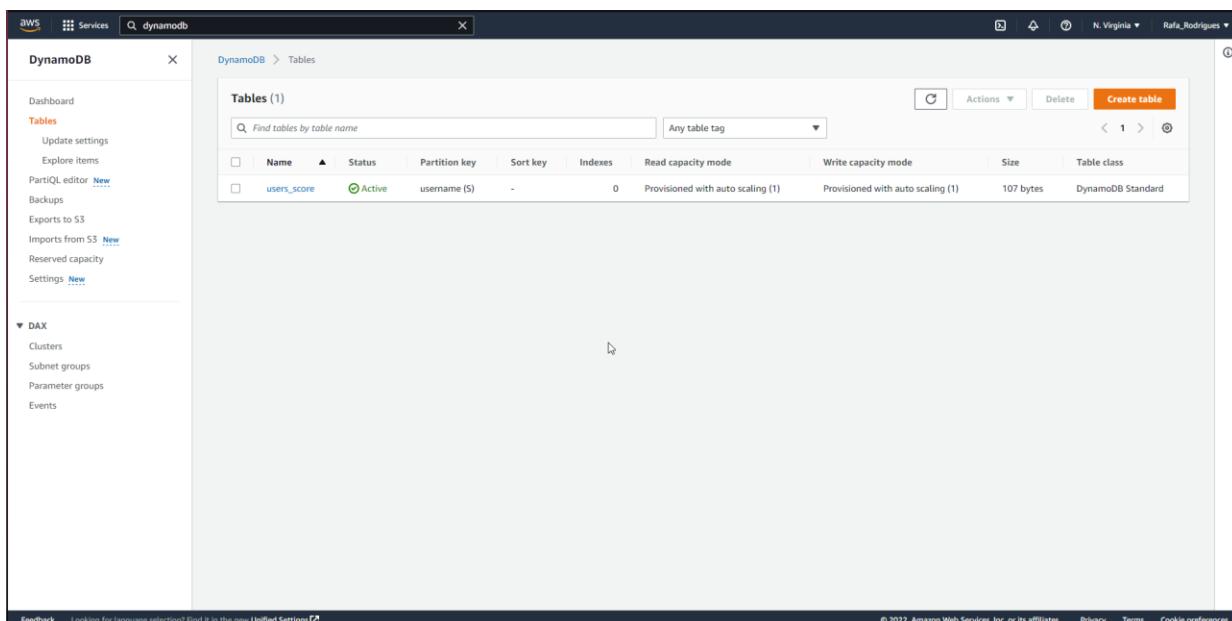
Link para a documentação oficial da APL: <https://developer.amazon.com/en-US/docs/alexa/alexa-design/apl.html#:~:text=Alexa%20has%20a%20visual%20design,an%20engaging%20to%20the%20customer>.

## DynamoDB

A skill está utilizando um servidor hospedado na Lambda com isso alguns recursos da Alexa não estão disponíveis, desta forma se faz necessário criar permissões de uso para a skill ter acesso a outras funcionalidades uma delas seria a persistência dos dados, neste caso será utilizado outro serviço da Amazon chamado de DynamoDB.

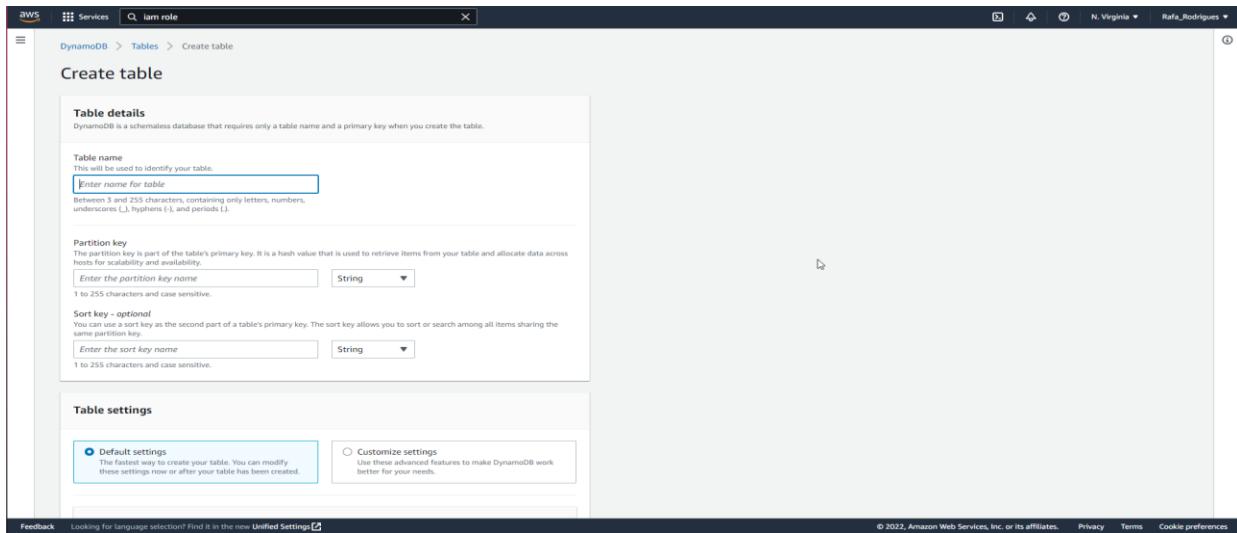
Em suma, o que será feito é: criar um usuário com permissão de acesso ao DynamoDB, criar uma tabela, adicionar a permissão na função.

Figura 42 - Página inicial DynamoDB



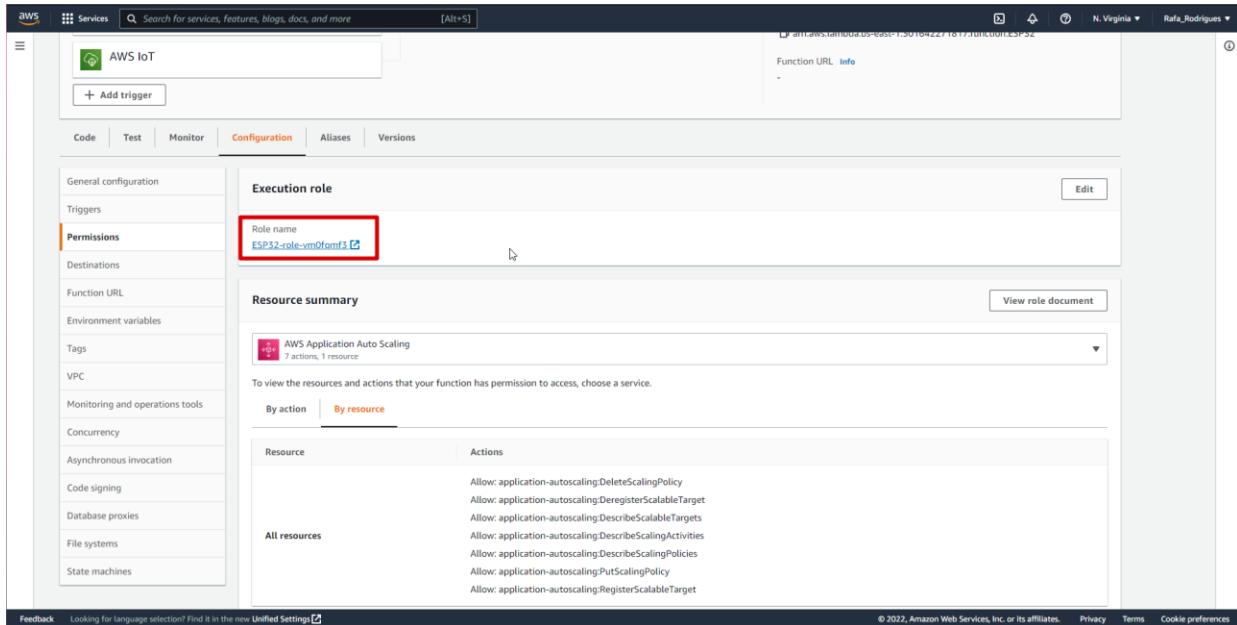
Criar a tabela informando o nome desejado.

Figura 43 - Criação de tabela



No Lambda clicar no hiperlink da permissão para abrir a página de edição.

Figura 44 - Permissão



Deve ser adicionado a permissão para acesso ao DynamoDB, desta forma clica na opção “Add permissions -> Attach policies”.

Figura 45 - Página de permissões

The screenshot shows the AWS IAM Roles page. The left sidebar has 'Identity and Access Management (IAM)' selected. The main area shows the 'ESP32-role-vm0fqmf3' role. Under the 'Permissions' tab, there are two attached policies listed:

- AWSLambdaBasicExecutionRole
- AmazonDynamoDBFullAccess

At the top right of the 'Permissions' section, there is a button labeled 'Attach policies' with a red box around it.

Procurar pela política “AmazonDynamoDBFullAccess” e clicar na opção “Attach policies”.

Figura 46 - Adicionando nova permissão

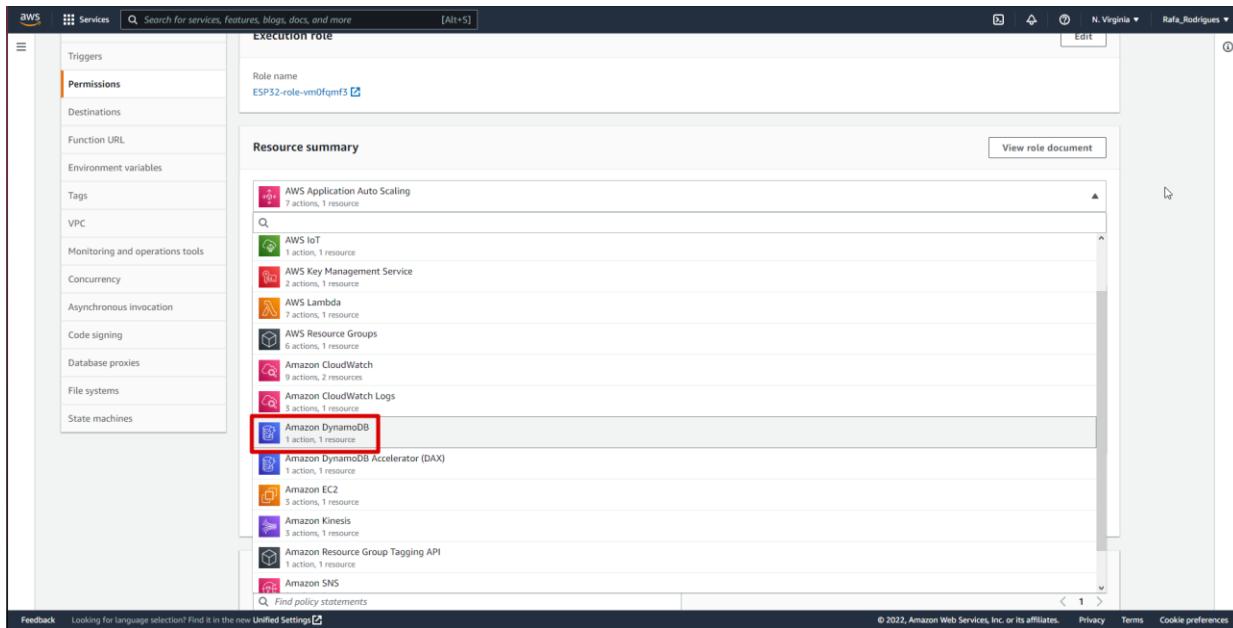
The screenshot shows the 'Attach policy to ESP32' dialog. It lists one current permission policy and many other available policies. The 'AmazonDynamoDBFullAccess' policy is highlighted with a red box.

Policy name	Type	Description
AmazonDynamoDBFullAccess	AWS managed	Provides full access to Amazon DynamoDB via the AWS Management Console.
AWSLambdaDynamoDBExecutionRole	AWS managed	Provides list and read access to DynamoDB streams and write permissions to CloudWatch logs.
AmazonDynamoDBReadOnlyAccess	AWS managed	Provides read only access to Amazon DynamoDB via the AWS Management Console.
AWSLambdaInvocation-DynamoDB	AWS managed	Provides read access to DynamoDB Streams.

At the bottom right of the dialog, there is a 'Cancel' button and a 'Attach policies' button.

Voltar a página do Lambda, e verificar se será apresentado na lista de permissões o DynamoDB.

Figura 47 - Lista de permissões



### Exemplos de utilização DynamoDB

Neste item será mostrado alguns exemplos de uso utilizando o DynamoDB, os exemplos são de inserir, atualizar, consultar e excluir dados. Esses exemplos estão disponibilizados no github e são baseados na documentação oficial que pode ser encontrado no site abaixo:

<https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/dynamodb.html#dynamodb>

#### Inserir

Exemplo para inserir um novo registro na tabela no Quadro 2.

Quadro 2 - Insert

```
--INSERT

#importing packages
import json
import boto3
#function definition
def lambda_handler(event,context):
    dynamodb=boto3.resource('dynamodb')
    #table name
    table = dynamodb.Table('users_score')
    #inserting values into table
    response = table.put_item(
        Item={
            'username': 'Teste',
            'score':3000
        }
    )
    return response
```

### *Atualizar*

Exemplo para atualizar um dado na tabela no Quadro 3.

*Quadro 3 - Update*

```
--UPDATE

#importing packages
import json
import boto3
#function definition
def lambda_handler(event,context):
    dynamodb=boto3.resource('dynamodb')
    #table name
    table = dynamodb.Table('users_score')
    #inserting values into table
    response = table.update_item(
        Key={"username":"Teste"},
        UpdateExpression="SET score= :s",
        ExpressionAttributeValues={':s':100},
        #Só precisa desse se for utilizar o response
        ReturnValues="UPDATED_NEW"
    )
    return response['Attributes']
```

### *Consultar*

Exemplo para retornar um dado na tabela no Quadro 4.

*Quadro 4 - Select*

```
--GET

#importing packages
import json
import boto3
#function definition
def lambda_handler(event,context):
    dynamodb=boto3.resource('dynamodb')
    #table name
    table = dynamodb.Table('users_score')
    users = table.scan()['Items']
    users_test = ""
    for users_l in users:
        if "bhaagiii" == users_l['username']:
            users_test += users_l['username'] + " - > " + users_l['score']
    print(users_test)
    return None
```

### *Excluir*

Exemplo para excluir um dado na tabela no Quadro 5.

*Quadro 5 - Delete*

```
--DELETE

#importing packages
import json
import boto3
#function definition
def lambda_handler(event,context):
```

```

dynamodb=boto3.resource('dynamodb')
#table name
table = dynamodb.Table('users_score')
#inserting values into table
response = table.delete_item(
    Key={"username":"Teste"},
    ReturnValues="ALL_OLD"
)
return response['Attributes']

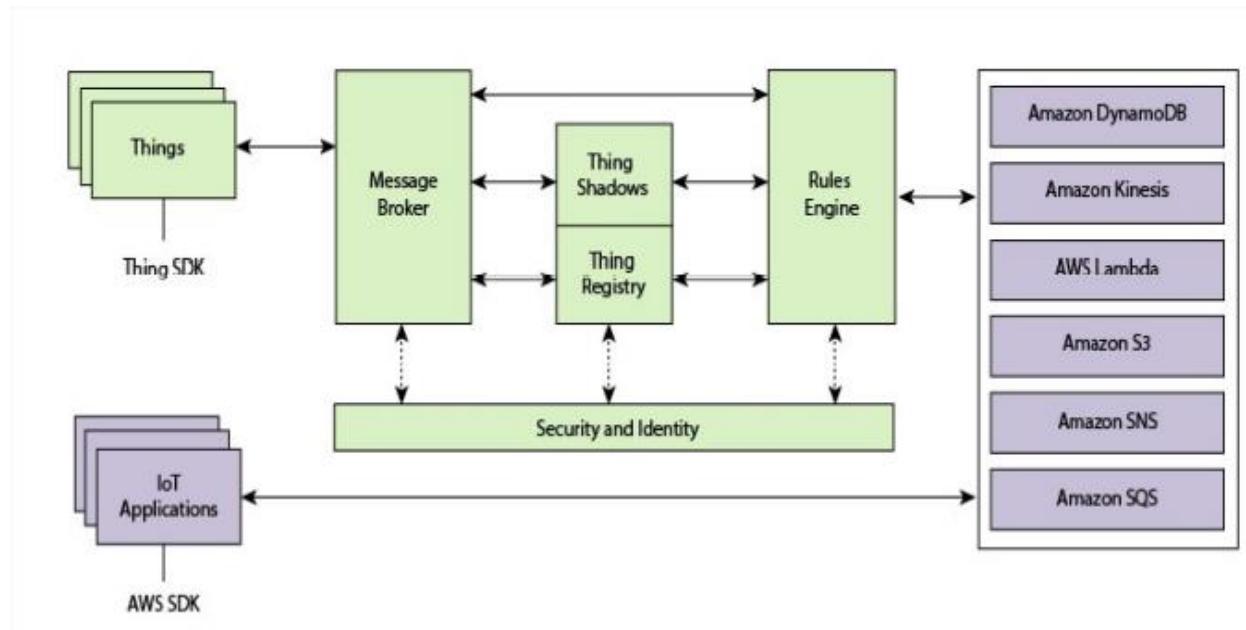
```

#### Criação de uma “coisa”

Conforme AMEBA, na arquitetura, ESP32 pertence ao bloco "Coisas" superior esquerdo. Um canal seguro TLS será estabelecido entre "Things" e o MQTT Message Broker. Em seguida, "coisa" e "Message Broker" se comunicam usando o Protocolo MQTT por meio desse canal seguro. Atrás do "Message Broker", as "Thing Shadows" mantêm mensagens temporariamente quando o ESP32 está offline, e envia a mensagem de controle para o ESP32 na próxima vez que ele estiver conectado. O "Mecanismo de Regras" permite que você coloque restrições ao comportamento das Coisas ou conecte Coisas a outros serviços da Amazon.

Na Figura 48 se encontra uma exemplificação desta arquitetura.

*Figura 48 - Estrutura do AWS IoT*



A partir da Figura 49 está descrito o passo-a-passo de como criar uma “coisa”, ou um novo dispositivo, à função em AWS IoT, terminando na Figura X.

Em suma, os passos são os seguintes: Registro no Console do AWS IoT; Criação/registro de uma "Coisa" no console; Criação de um certificado TLS X.509; Criação de uma política para acessar os tópicos da AWS IoT; Anexando a Política ao Certificado; testes via MQTT.

Passo 1: Acessar a página do IoT Core, acessar o item “Things” e utilizar a opção “Create things”.

Figura 49 - Página inicial AWS IoT

The screenshot shows the AWS IoT Things page. On the left, there's a navigation sidebar with options like Monitor, Connect, Test, and Manage. Under Manage, 'All devices' is expanded, and 'Things' is selected, which is highlighted with a red box. The main content area shows a table with columns 'Name' and 'Thing type'. A message at the top says 'No things' and 'No things to display in this Region'. At the bottom right of the table, there's a 'Create things' button, also highlighted with a red box. The top right corner shows the user 'Rafa\_Rodrigues'.

Passo 2: Utilizar a opção “Create single thing” e clicar em “Next”.

Figura 50 - Criação da “coisa”

The screenshot shows the 'Create things' wizard. The first step, 'Number of things to create', has two options: 'Create single thing' (selected) and 'Create many things'. Below each option is a brief description. At the bottom right of the step, there are 'Cancel' and 'Next' buttons, with 'Next' being highlighted with a red box. The top right corner shows the user 'Rafa\_Rodrigues'.

Passo 3: Inserir um novo nome e demais valores são opcionais.

Figura 51 - Propriedades da “coisa”

A thing resource is a digital representation of a physical device or logical entity in AWS IoT. Your device or entity needs a thing resource in the registry to use AWS IoT features such as Device Shadows, events, jobs, and device management features.

**Thing properties**

Thing name  
ESP32\_SP  
Enter a unique name containing only letters, numbers, hyphens, colons, or underscores. A thing name can't contain any spaces.

Additional configurations

You can use these configurations to add detail that can help you to organize, manage, and search your things.

- Thing type - optional
- Searchable thing attributes - optional
- Thing groups - optional
- Billing group - optional

**Device Shadow**

Device Shadows allow connected devices to sync states with AWS. You can also get, update, or delete the state information of this thing's shadow using either HTTPS or MQTT topics.

- No shadow
- Named shadow  
Create multiple shadows with different names to manage access to properties, and logically group your devices properties.
- Unnamed shadow (classic)  
A thing can have only one unnamed shadow.

Cancel **Next**

Passo 4: Utilizar a opção “Auto-generate a new certificate (recommended)” e clicar em “Next”.

Figura 52 - Escolha do certificado

AWS IoT > Manage > Things > Create things > Create single thing

**Configure device certificate - optional**

A device requires a certificate to connect to AWS IoT. You can choose how you to register a certificate for your device now, or you can create and register a certificate for your device later. Your device won't be able to connect to AWS IoT until it has an active certificate with an appropriate policy.

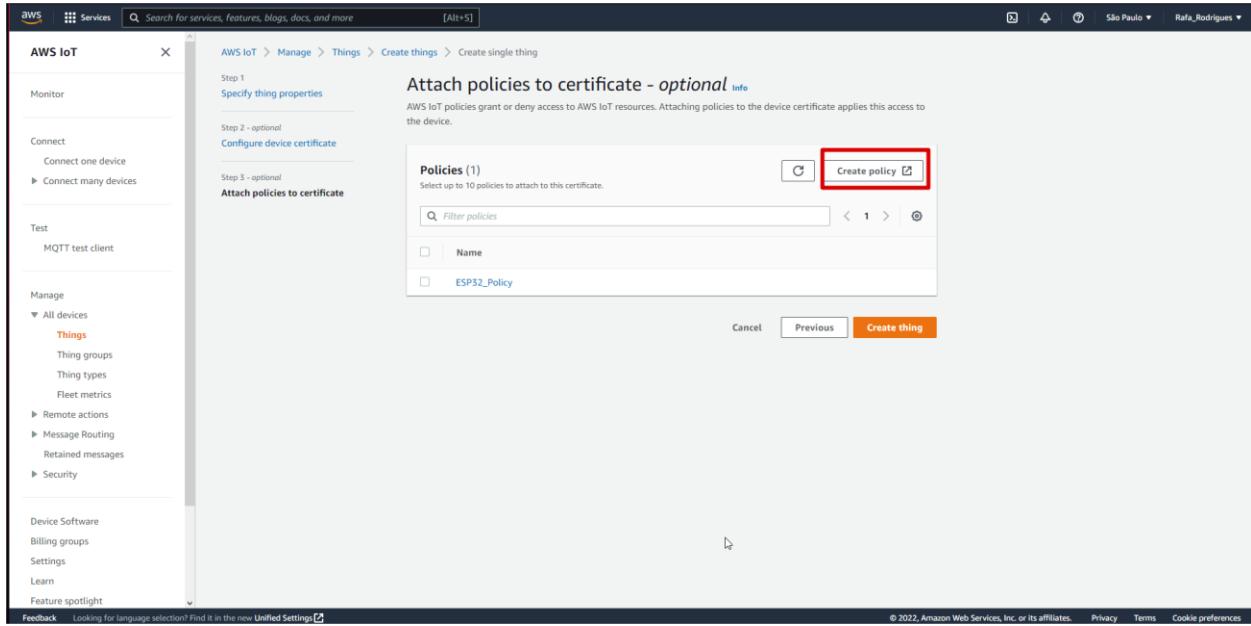
**Device certificate**

- Auto-generate a new certificate (recommended)  
Generate a certificate, public key, and private key using AWS IoT's certificate authority.
- Use my certificate  
Use a certificate signed by your own certificate authority.
- Upload CSR  
Register your CA and use your own certificates on one or many devices.
- Skip creating a certificate at this time  
You can create a certificate for this thing and attach a policy to the certificate at a later time.

Cancel **Previous** **Next**

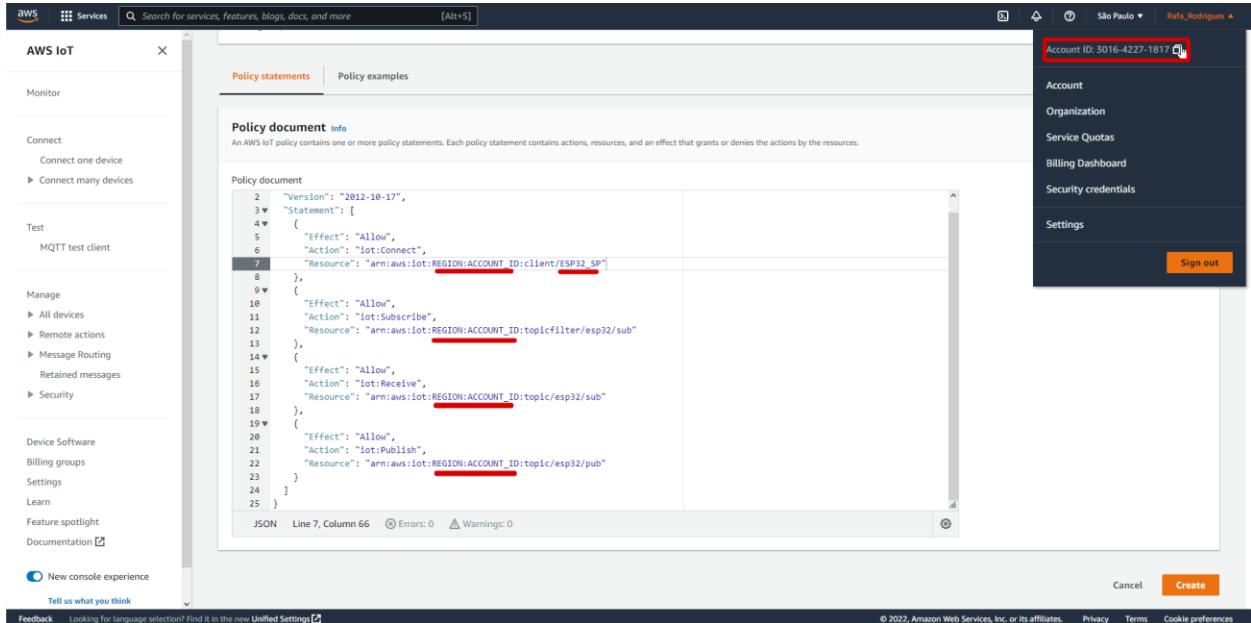
## Passo 5: Criar uma política.

Figura 53 - Política da “coisa”



Passo 6: Inserir o código do Quadro 1 e alterar com os dados de região, ID da conta e alterar o nome ESP32\_SP para o nome da “coisa”.

Figura 54 - Criação de política



Quadro 6 - Código da política

### Código política

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iot:Connect",
      "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:client/ESP32_SP"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Subscribe",
      "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/esp32/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Receive",
      "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topic/esp32/sub"
    },
    {
      "Effect": "Allow",
      "Action": "iot:Publish",
      "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topic/esp32/pub"
    }
  ]
}
```

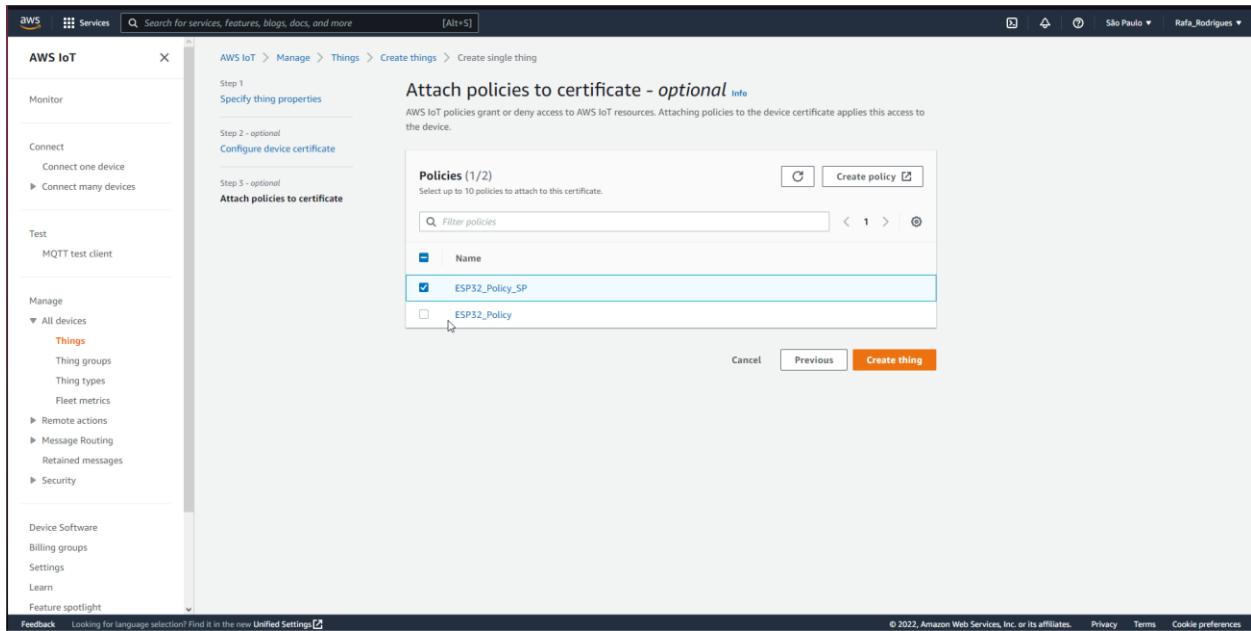
```

    "Action": "iot:Connect",
    "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:client/MyNewESP32"
},
{
    "Effect": "Allow",
    "Action": "iot:Subscribe",
    "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topicfilter/esp32/sub"
},
{
    {
        "Effect": "Allow",
        "Action": "iot:Receive",
        "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topic/esp32/sub"
    },
    {
        "Effect": "Allow",
        "Action": "iot:Publish",
        "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topic/esp32/pub"
    }
]
}

```

#### Passo 7: Vincular a política e criar a “coisa”

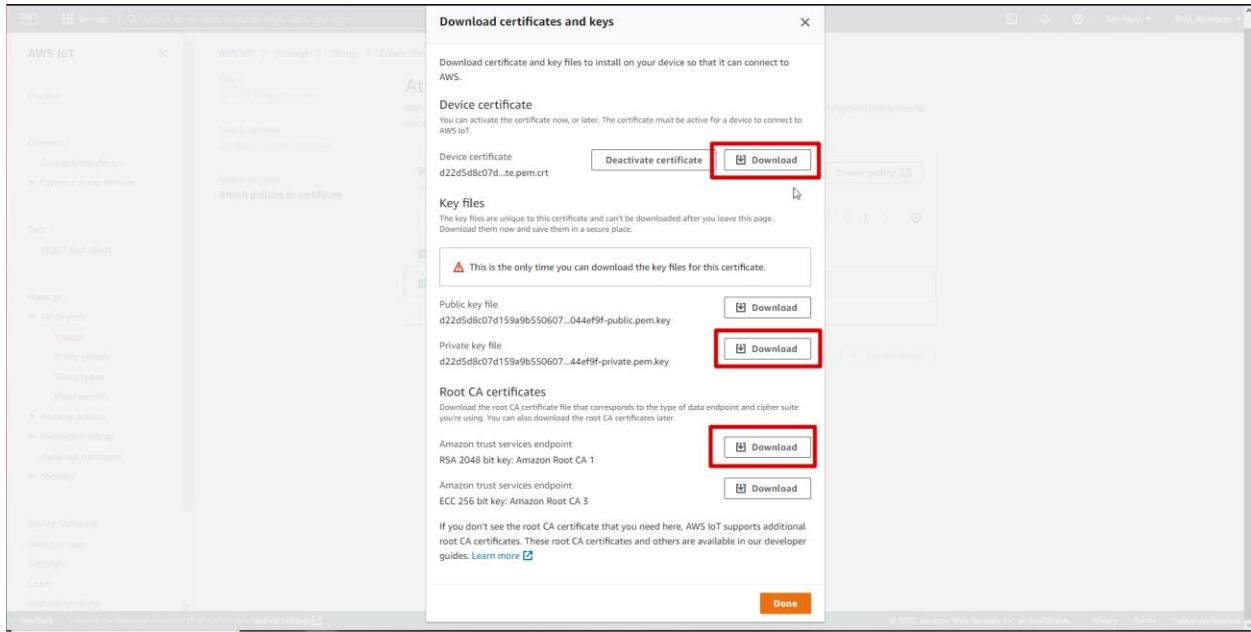
*Figura 55 - Vínculo da política*



Passo 8: Download de todos os certificados, esses certificados são os citados dentro do item “Secrets.h” para serem inseridos no código do Arduino.

Obs.: No código do Arduino consta um comentário onde deve ser inserido cada conteúdo dos arquivos baixados.

Figura 56 - Download dos certificados



## Testes MQTT

Nas subseções será apresentado como realizar testes via MQTT dentro do serviço IoT Core.

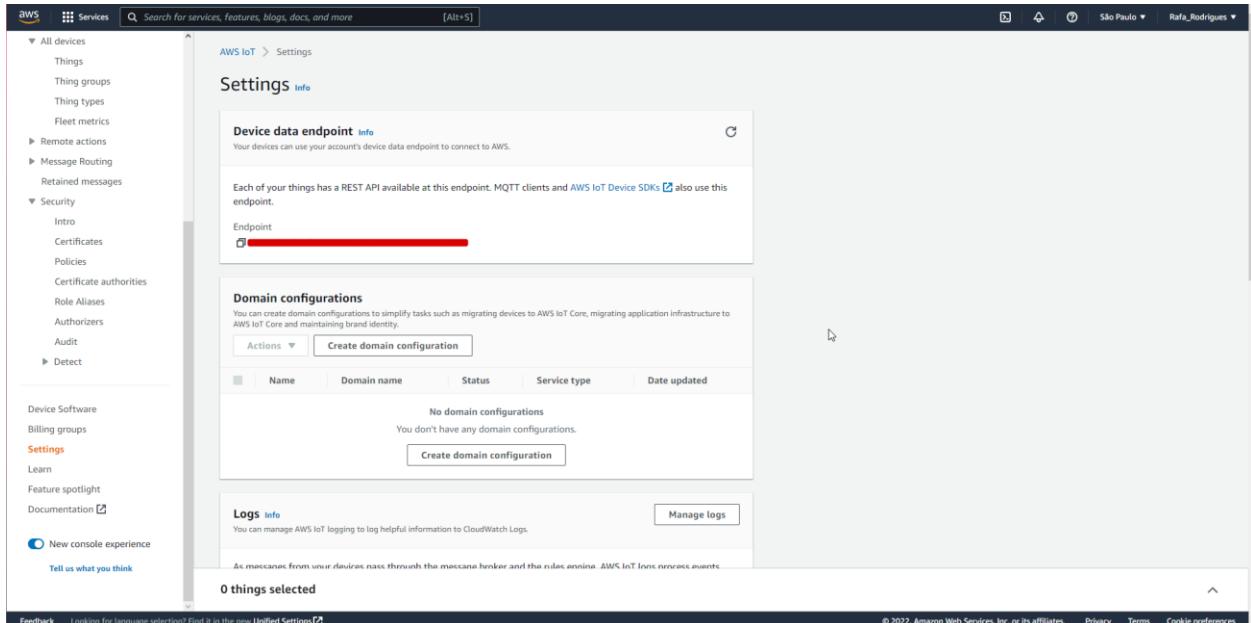
### Variáveis

Entrar no arquivo “Secrets.h” do Arduino e atualizar as seguintes informações:

- THINGNAME: Nome da “coisa” criada.
- AWS\_IOT\_ENDPOINT[] : Este valor pode ser encontrado dentro do item “Settings” da página do IoT Core, conforme Figura 57.
- AWS\_CERT\_CA[],AWS\_CERT\_CRT[] e AWS\_CERT\_PRIVATE[]: Conteúdo se encontra dentro dos arquivos baixados ao criar a “coisa”.

Após isso compilar e utilizar a opção “Upload” para carregar o código dentro da memória do ESP32.

Figura 57 – Endpoint IoT Core



## Ferramenta de testes MQTT Lambda

Após acessar a página do IoT Core, acessar o item “MQTT test client”. E configurar os tópicos conforme Figura 58 e Figura 59.

Figura 58 - Configuração do tópico MQTT

AWS IoT > MQTT test client

MQTT test client [Info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Subscribe to a topic [Publish to a topic](#)

Topic filter [Info](#)  
The topic filter describes the topic(s) to which you want to subscribe. The topic filter can include MQTT wildcard characters.  
esp32/pub

Additional configuration

Subscribe

Subscriptions

esp32/pub

No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

Pause Clear Export Edit

Feedback Looking for language selection? Find it in the new Unified Settings [\[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Figura 59 - Configuração do tópico MQTT

AWS IoT > MQTT test client

MQTT test client [Info](#)

You can use the MQTT test client to monitor the MQTT messages being passed in your AWS account. Devices publish MQTT messages that are identified by topics to communicate their state to AWS IoT. AWS IoT also publishes MQTT messages to inform devices and apps of changes and events. You can subscribe to MQTT message topics and publish MQTT messages to topics by using the MQTT test client.

Subscribe to a topic [Publish to a topic](#)

Topic name  
The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.  
esp32/sub

Message payload

```
{  
  "message": "Hello from AWS IoT console"  
}
```

Additional configuration

Publish

Subscriptions

esp32/pub

No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.

Pause Clear Export Edit

Feedback Looking for language selection? Find it in the new Unified Settings [\[Feedback\]](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Para realizar o teste basta inserir uma nova mensagem e utilizar a opção “Publish”, dentro do Arduino é possível ver os logs pelo canal escolhido e posteriormente a isto será enviado uma mensagem de sucesso novamente para a Amazon. Segue figuras com esse processo.

Figura 60 - Envio da mensagem ao Arduino

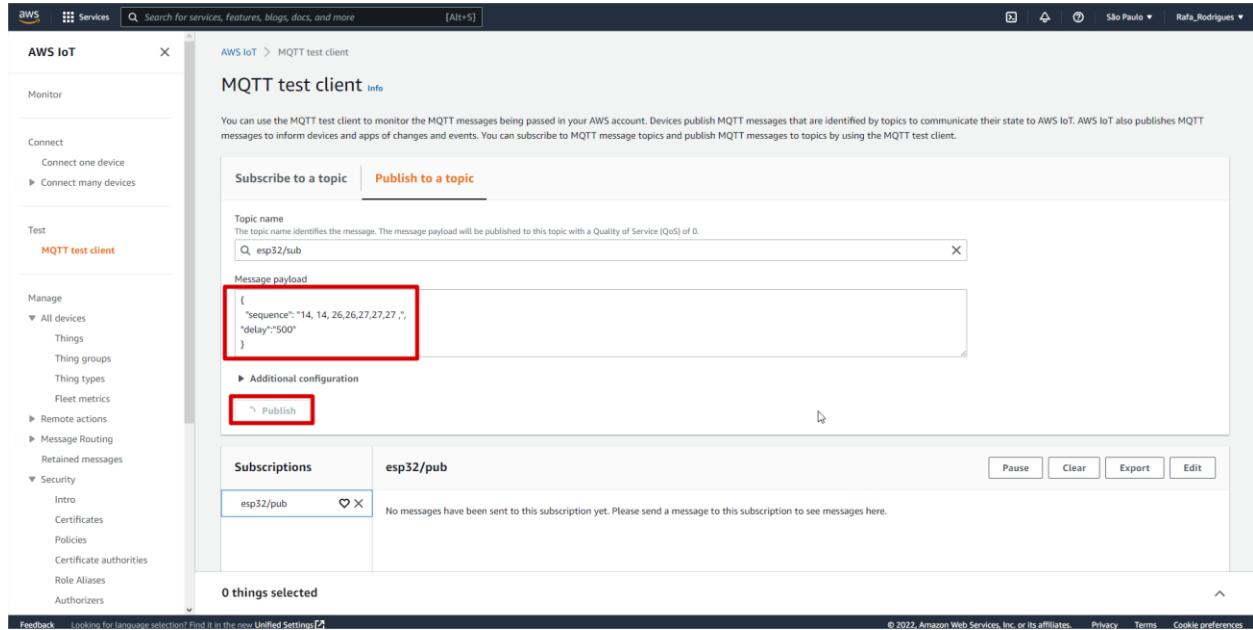


Figura 61 - Log de recebimento no Arduino

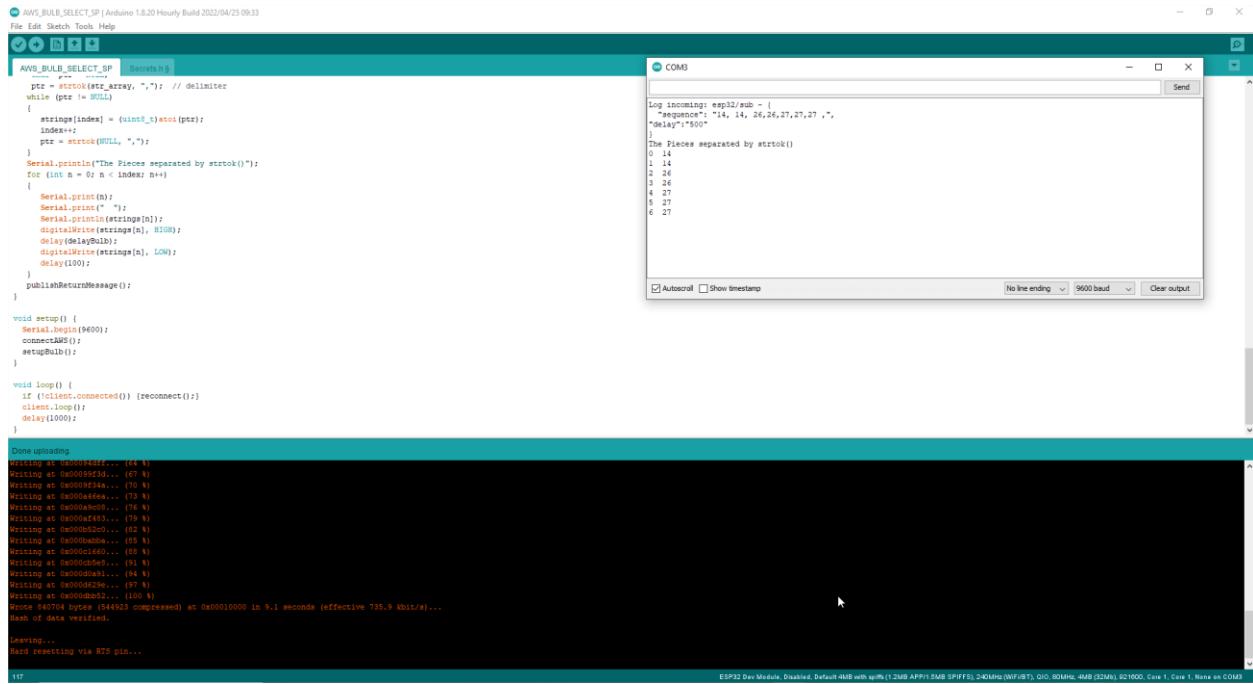
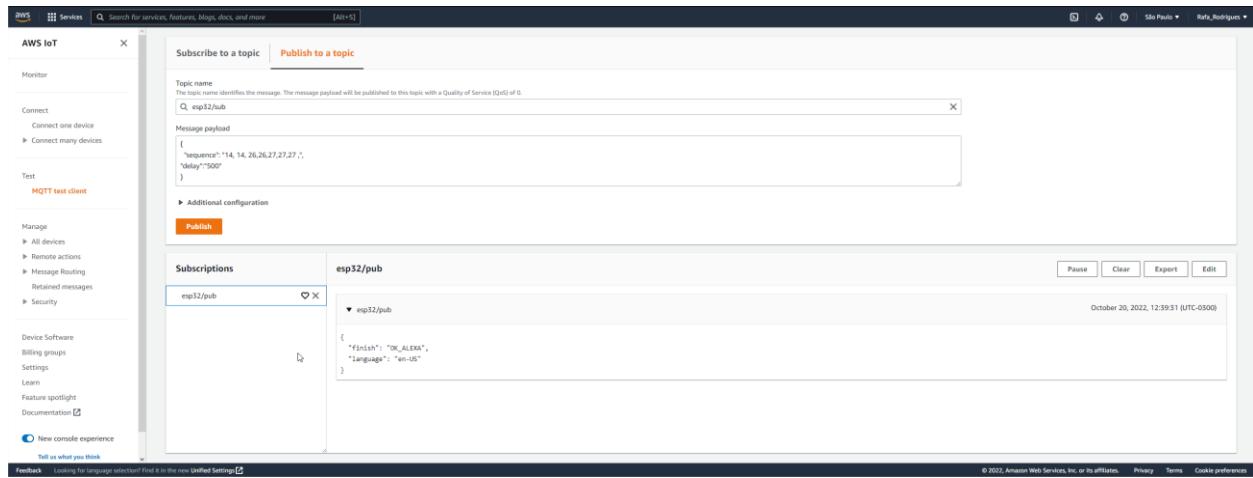


Figura 62 - Log de sucesso IoT Core



## Logs CloudWatch

Por padrão ao se criar uma skill e essa esteja na AWS Lambda ou dentro do servidor criado automaticamente pela Amazon, é vinculado o serviço CloudWatch que gera os logs da sessão da skill sejam erro ou de execução, ou até mesmo logs manuais inseridos dentro do código fonte.

Para logs manuais, em Python, é necessário importar a biblioteca logger, criar uma instância e o nível do log a ser gerado. Segue no Quadro 7.

Quadro 7 - Como criar um log

```
#Biblioteca
import logging

#Criar uma nova instância
logger = logging.getLogger(__name__)
logger.setLevel(logging.INFO)

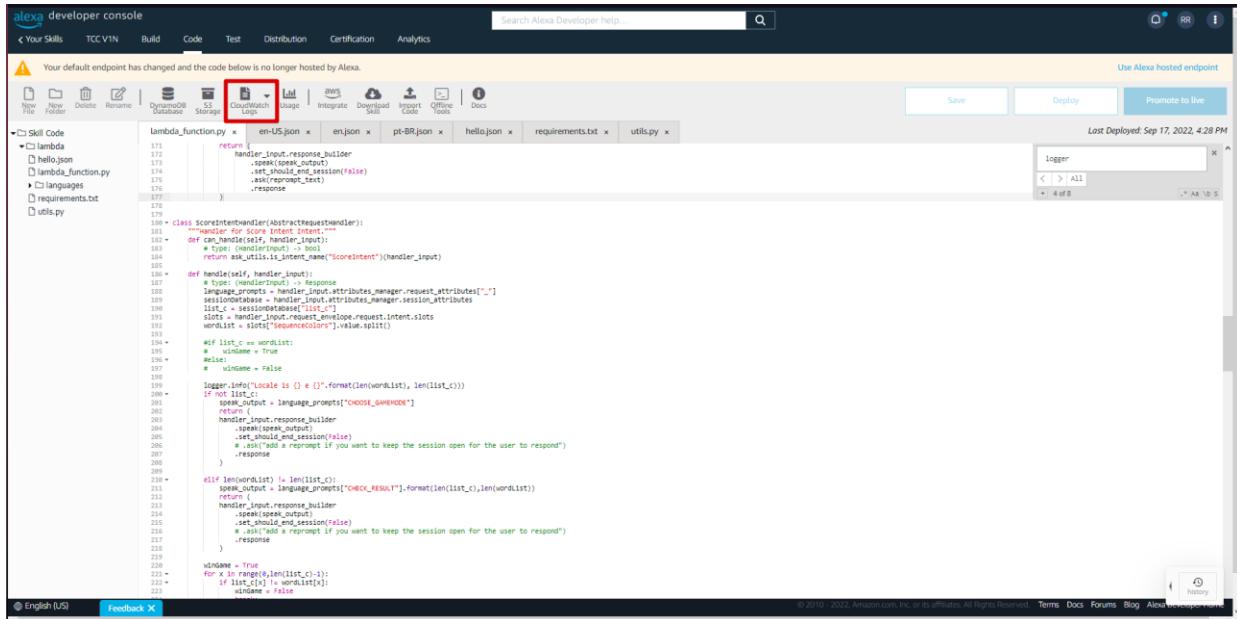
#Exemplo de uso
logger.info("Locale is {} e {}".format(len(wordList), len(list_c)))
```

Desta forma, é possível acessar esse serviço de duas formas, conforme será mostrado nas subseções a seguir.

## Servidor hospedado na Alexa

Caso seja servidor hospedado na própria Alexa, ir na aba “Code” e clicar no ícone “CloudWatch Logs”.

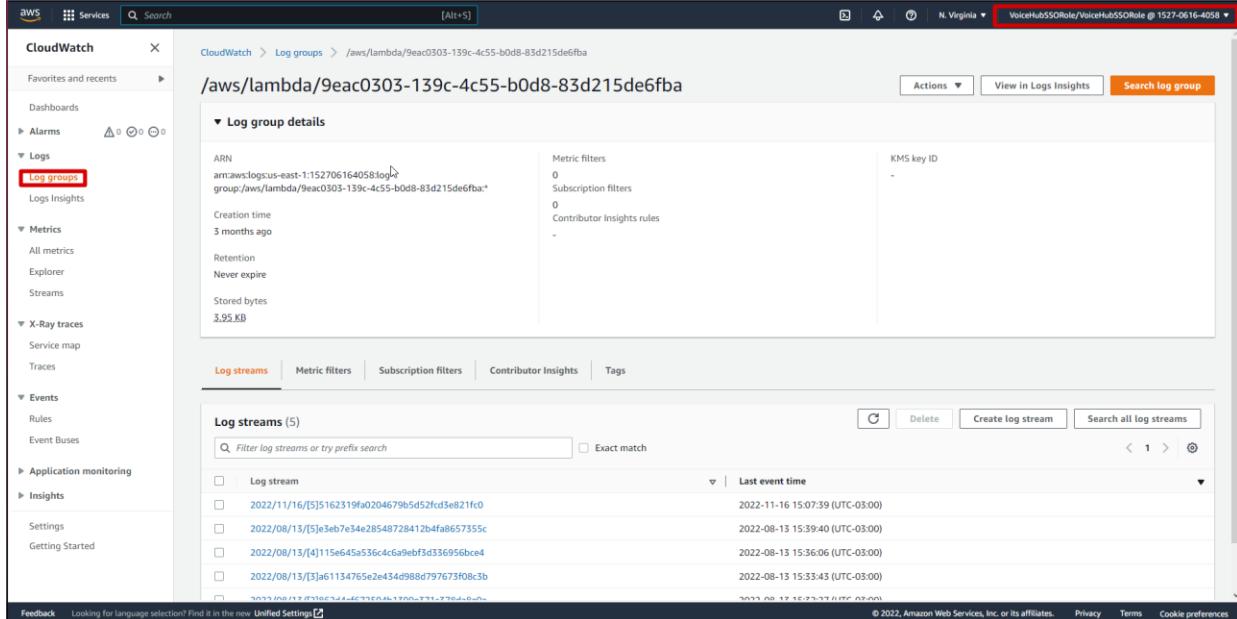
Figura 63 - Ícone CloudWatch Logs



Será aberta a página da CloudWatch, nela terá que selecionar o item “Log groups”, e selecionar algum registro. Esse registro constará com o que foi feito na seção, contendo o número da sessão, tempo inicial e final sendo possível visualizar em tempo real.

Obs.: Na parte superior a direita é possível ver um usuário com o prefixo “VoiceHubSSORole” esse usuário é gerado pela própria skill e não temos controle dele ou das permissões dele.

Figura 64 - CloudWatch item LogGroups



Por fim, um exemplo de como o log será gerado.

Figura 65 - Exemplo de log

The screenshot shows the AWS CloudWatch Log Groups interface. The left sidebar navigation includes CloudWatch, Favorites and recent, Dashboards, Alarms, Logs (selected), Log groups, Logs Insights, Metrics, X-Ray traces, Service map, Traces, Events, Rules, Event Buses, Application monitoring, and Insights. The main content area displays 'Log events' for the path /aws/lambda/9eac0305-159c-4c55-b0db-83d215de6fba. The logs are filtered by 'Timestamp' and 'Message'. The first few entries are:

- 2022-08-13T15:33:34.831+03:00 OpenBLAS WARNING - could not determine the L2 cache size on this system, assuming 256K
- 2022-08-13T15:33:34.971+03:00 START RequestId: b5c1743d-1bd7-407e-8696-87fc34d9e03 Version: 3
- 2022-08-13T15:33:35.027+03:00 END RequestId: b5c1743d-1bd7-407e-8696-87fc34d9e03
- 2022-08-13T15:33:35.027+03:00 REPORT RequestId: b5c1743d-1bd7-407e-8696-87fc34d9e03 Duration: 54.81 ms Billed Duration: 55 ms Memory Size: 512 MB Max Memory Used: 70 MB Init Duration: 499.56 ms
- 2022-08-13T15:33:38.519+03:00 START RequestId: ee95ad5a-f907-4045-9a24-24a7ddcccf940 Version: 3
- 2022-08-13T15:33:38.519+03:00 [ERROR] 2022-08-13T18:33:38.518Z ee95ad5a-f907-4045-9a24-24a7ddcccf940 Unable to find a suitable request handler
- 2022-08-13T15:33:38.519+03:00 Traceback (most recent call last):
- 2022-08-13T15:33:38.519+03:00 File "/var/task/ask\_sdm\_runtime/dispatch.py", line 118, in dispatch
- 2022-08-13T15:33:38.519+03:00 output = self.\_dispatch\_request(handler\_input) # type: Union[Output, None]
- 2022-08-13T15:33:38.519+03:00 File "/var/task/ask\_sdm\_runtime/dispatch.py", line 165, in \_dispatch\_request
- 2022-08-13T15:33:38.519+03:00 "Unable to find a suitable request handler"
- 2022-08-13T15:33:38.519+03:00 ask\_sdm\_runtime.exceptions.DispatchException: Unable to find a suitable request handler
- 2022-08-13T15:33:38.520+03:00 END RequestId: ee95ad5a-f907-4045-9a24-24a7ddcccf940
- 2022-08-13T15:33:38.520+03:00 REPORT RequestId: ee95ad5a-f907-4045-9a24-24a7ddcccf940 Duration: 4.67 ms Billed Duration: 5 ms Memory Size: 512 MB Max Memory Used: 71 MB
- 2022-08-13T15:33:43.365+03:00 START RequestId: 2083e02e-7ae4-4756-af00-4acfd394d1c9 Version: 3
- 2022-08-13T15:33:43.365+03:00 [ERROR] 2022-08-13T18:33:43.388Z 2083e02e-7ae4-4756-af00-4acfd394d1c9 Unable to find a suitable request handler
- 2022-08-13T15:33:43.365+03:00 Traceback (most recent call last):
- 2022-08-13T15:33:43.365+03:00 File "/var/task/ask\_sdm\_runtime/dispatch.py", line 118, in dispatch
- 2022-08-13T15:33:43.365+03:00 output = self.\_dispatch\_request(handler\_input) # type: Union[Output, None]
- 2022-08-13T15:33:43.365+03:00 File "/var/task/ask\_sdm\_runtime/dispatch.py", line 165, in \_dispatch\_request
- 2022-08-13T15:33:43.365+03:00 "Unable to find a suitable request handler"
- 2022-08-13T15:33:43.365+03:00 ask\_sdm\_runtime.exceptions.DispatchException: Unable to find a suitable request handler
- 2022-08-13T15:33:43.369+03:00 END RequestId: 2083e02e-7ae4-4756-af00-4acfd394d1c9
- 2022-08-13T15:33:43.369+03:00 REPORT RequestId: 2083e02e-7ae4-4756-af00-4acfd394d1c9 Duration: 3.07 ms Billed Duration: 4 ms Memory Size: 512 MB Max Memory Used: 71 MB

No newer events at this moment. Auto retry paused. Resume

## Servidor hospedado na AWS Lambda

Para verificar os logs gerados em uma skill hospedado na AWS Lambda, é necessário apenas procurar o serviço na barra de procurar e novamente entrar no item “Log Groups” que será apresentado os logs gerados na sessão ao realizar um teste pelo console do desenvolvedor ou pela própria Alexa.

Figura 66 - Exemplo de log lambda

The screenshot shows the AWS CloudWatch Log Groups interface. The left sidebar navigation includes CloudWatch, Favorites and recent, Dashboards, Alarms, Logs (selected), Log groups, Logs Insights, Metrics, X-Ray traces, Service map, Traces, Events, Rules, Event Buses, Application monitoring, and Insights. The main content area displays 'Log events' for the path /aws/lambda/ESP32. The logs are filtered by 'Timestamp' and 'Message'. The first few entries are:

- 2022-11-16T15:17:33.186+03:00 START RequestId: cb66c40b-c8ee-4059-0551-0a2266c8c31e Version: \$LATEST
- 2022-11-16T15:17:33.391+03:00 END RequestId: cb66c40b-c8ee-4059-0551-0a2266c8c31e
- 2022-11-16T15:17:33.391+03:00 REPORT RequestId: cb66c40b-c8ee-4059-0551-0a2266c8c31e Duration: 204.96 ms Billed Duration: 205 ms Memory Size: 128 MB Max Memory Used: 61 MB Init Duration: 746.93 ms
- 2022-11-16T15:17:40.249+03:00 START RequestId: 04de1b32-ed0d-4aef-ba01-98d0a954256e Version: \$LATEST
- 2022-11-16T15:17:40.269+03:00 END RequestId: 04de1b32-ed0d-4aef-ba01-98d0a954256e
- 2022-11-16T15:17:40.269+03:00 REPORT RequestId: 04de1b32-ed0d-4aef-ba01-98d0a954256e Duration: 20.62 ms Billed Duration: 21 ms Memory Size: 128 MB Max Memory Used: 61 MB

>Loading newer events

## Informações adicionais

Nos tópicos a seguir, seriam informações interessantes de ter em mente ao desenvolver uma skill.

### Internacionalização

Na computação, a internacionalização é o processo de projetar software para que possa ser adaptado a diferentes idiomas e regiões sem a necessidade de reescrever ou alterar o código fonte.

A internacionalização permite que você crie uma skill para a Alexa que suporte vários idiomas, mas todos os idiomas irão utilizar o mesmo servidor e consequentemente o mesmo código fonte. Quando o usuário faz uma solicitação, parte do JSON de entrada contém as informações de localidade do dispositivo que está usando. Sua skill pode usar essas informações para falar no idioma apropriado para a solicitação do usuário.

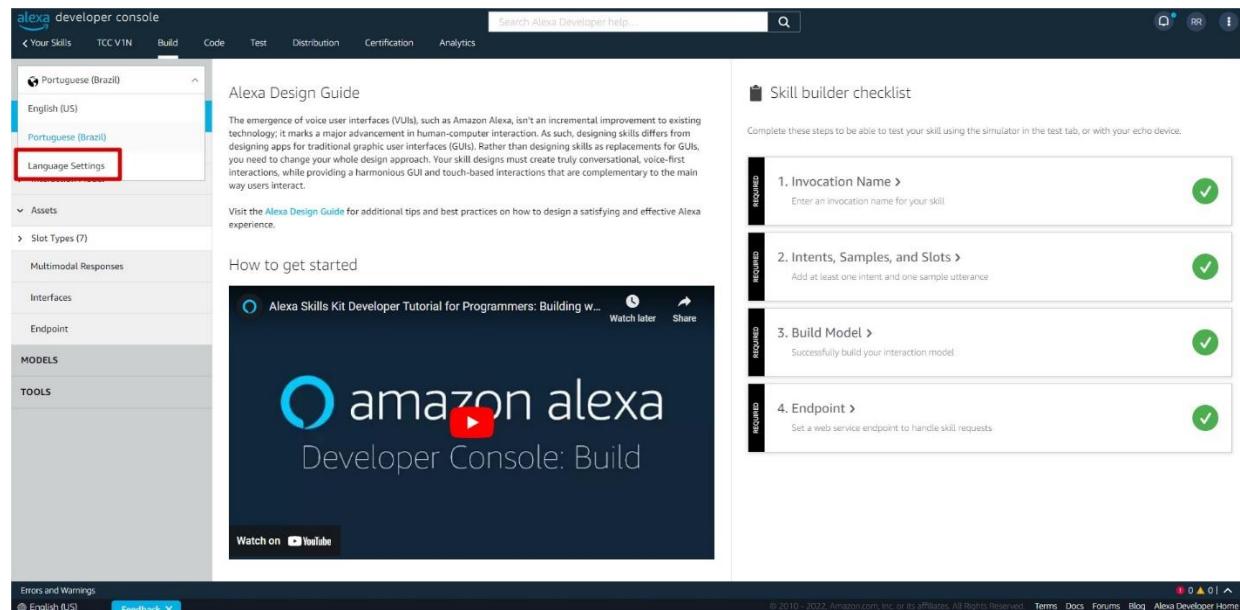
Existem duas formas possíveis para se realizar isso, a primeira seria identificar o valor do JSON e toda vez executar um "IF" para conseguir utilizar o idioma de entrada e retornasse a resposta do idioma adequado. Para skills extensas não seria ideal pois toda alteração poderia gerar uma grande manutenção, um grande esforço e mesmo assim poderia ocorrer erros.

Outra consideração ao pensar em uma estratégia de internacionalização para grandes skills é terceirizar a tradução para outra empresa. Obviamente, você não deseja enviar a eles seu código fonte para que eles possam atualizá-lo; portanto, as sequências de texto geralmente são colocadas em um arquivo separado que você pode enviar à empresa de tradução para edição.

### Modelo de interação - Parte 1

O console do desenvolvedor da Alexa já consta com uma opção para criar um modelo de forma simples, para isso você deve abrir a lista de idiomas e clicar na opção "Language Settings", conforme figura 67.

Figura 67 - Lista de idiomas



Será aberta uma nova página, nesta página deve selecionar "Add new language", escolher o idioma desejado e clicar em "Save". Após salvar, será apresentado na lista o novo idioma adicionado. Na figura 68 é apresentado botão para adicionar um novo idioma e salvar.

Figura 68 - Configurações de idiomas

The screenshot shows the Alexa developer console interface. In the top navigation bar, 'Your Skills' is selected. The main area is titled 'Language settings'. It lists two primary languages: 'English' and 'Portuguese'. Under 'Portuguese', there is a button labeled '+ Add new language' with a red box around it. To the right of the language names, there are sections for 'Sync Locales' and 'ACTIONS' (Clone | Delete). At the bottom right of the page, there is a 'Saved' button with a red box around it.

Para ficar simples e não ser necessário adicionar novamente todas as intenções, slots etc. Você pode entrar no idioma onde consta já todas as configurações, ir à opção "Intents -> JSON Editor", copiar o conteúdo do json e traduzir manualmente as frases para então entrar no novo idioma e colar esse JSON traduzido e salvar esse novo modelo fazendo todos os registros de forma automática. Na figura 69 é apresentado onde se pode encontrar o JSON.

Figura 69 - Item JSON Editor

The screenshot shows the Alexa developer console interface. In the top navigation bar, 'Your Skills' is selected. The left sidebar has several tabs: 'CUSTOM' (selected), 'Interaction Model', 'Intents (11)' (selected), 'Assets', 'Slot Types (7)', 'Multimodal Responses', 'Interfaces', 'Endpoint', 'MODELS', and 'TOOLS'. The 'Intents (11)' tab is currently active. The main area is titled 'JSON Editor' with the sub-instruction 'Click here to learn more about the schema definition for interaction models.' Below this is a code editor window containing a JSON file. The JSON code is as follows:

```

1 < 4   "interactionModel": {
2     "languageCode": "pt-BR",
3     "invocationName": "jogo da memória",
4     "intents": [
5         {
6             "name": "AMAZON.CancelIntent",
7             "samples": []
8         },
9         {
10            "name": "AMAZON.HelpIntent",
11            "samples": []
12        },
13        {
14            "name": "AMAZON.StopIntent",
15            "samples": []
16        },
17        {
18            "name": "AMAZON.NavigateHomeIntent",
19            "samples": []
20        },
21        {
22            "name": "InitialGameIntent",
23            "slots": [
24                {
25                    "name": "difficulty",
26                    "type": "Difficulty",
27                    "samples": [
28                        "(difÍculDifícil)",
29                        "(difÍculDifícil)",
30                        "(difÍculDifícil)",
31                        "(difÍculDifícil)",
32                        "(difÍculDifícil)",
33                        "(difÍculDifícil)"
34                    ],
35                    "values": [
36                        "(difÍculDifícil)",
37                        "(difÍculDifícil)",
38                        "(difÍculDifícil)",
39                        "(difÍculDifícil)"
40                    ]
41                }
42            ],
43            "samples": [
44                "(difÍculDifícil)"
45            ]
46        }
47    ],
48    "genotypes": [
49        {
50            "name": "genotypes",
51            "type": "Genotypes",
52            "samples": [
53                "(geno1 geno2 geno3 geno4 geno5 geno6 geno7 geno8 geno9 geno10 geno11 geno12 geno13 geno14 geno15 geno16 geno17 geno18 geno19 geno20 geno21 geno22 geno23 geno24 geno25 geno26 geno27 geno28 geno29 geno30 geno31 geno32 geno33 geno34 geno35 geno36 geno37 geno38 geno39 geno40 geno41 geno42 geno43 geno44 geno45 geno46 geno47 geno48 geno49 geno50 geno51 geno52 geno53 geno54 geno55 geno56 geno57 geno58 geno59 geno60 geno61 geno62 geno63 geno64 geno65 geno66 geno67 geno68 geno69 geno70 geno71 geno72 geno73 geno74 geno75 geno76 geno77 geno78 geno79 geno80 geno81 geno82 geno83 geno84 geno85 geno86 geno87 geno88 geno89 geno90 geno91 geno92 geno93 geno94 geno95 geno96 geno97 geno98 geno99 geno100 geno101 geno102 geno103 geno104 geno105 geno106 geno107 geno108 geno109 geno110 geno111 geno112 geno113 geno114 geno115 geno116 geno117 geno118 geno119 geno120 geno121 geno122 geno123 geno124 geno125 geno126 geno127 geno128 geno129 geno130 geno131 geno132 geno133 geno134 geno135 geno136 geno137 geno138 geno139 geno140 geno141 geno142 geno143 geno144 geno145 geno146 geno147 geno148 geno149 geno150 geno151 geno152 geno153 geno154 geno155 geno156 geno157 geno158 geno159 geno160 geno161 geno162 geno163 geno164 geno165 geno166 geno167 geno168 geno169 geno170 geno171 geno172 geno173 geno174 geno175 geno176 geno177 geno178 geno179 geno180 geno181 geno182 geno183 geno184 geno185 geno186 geno187 geno188 geno189 geno190 geno191 geno192 geno193 geno194 geno195 geno196 geno197 geno198 geno199 geno200 geno201 geno202 geno203 geno204 geno205 geno206 geno207 geno208 geno209 geno210 geno211 geno212 geno213 geno214 geno215 geno216 geno217 geno218 geno219 geno220 geno221 geno222 geno223 geno224 geno225 geno226 geno227 geno228 geno229 geno2210 geno2211 geno2212 geno2213 geno2214 geno2215 geno2216 geno2217 geno2218 geno2219 geno2220 geno2221 geno2222 geno2223 geno2224 geno2225 geno2226 geno2227 geno2228 geno2229 geno22210 geno22211 geno22212 geno22213 geno22214 geno22215 geno22216 geno22217 geno22218 geno22219 geno22220 geno22221 geno22222 geno22223 geno22224 geno22225 geno22226 geno22227 geno22228 geno22229 geno222210 geno222211 geno222212 geno222213 geno222214 geno222215 geno222216 geno222217 geno222218 geno222219 geno222220 geno222221 geno222222 geno222223 geno222224 geno222225 geno222226 geno222227 geno222228 geno222229 geno2222210 geno2222211 geno2222212 geno2222213 geno2222214 geno2222215 geno2222216 geno2222217 geno2222218 geno2222219 geno2222220 geno2222221 geno2222222 geno2222223 geno2222224 geno2222225 geno2222226 geno2222227 geno2222228 geno2222229 geno22222210 geno22222211 geno22222212 geno22222213 geno22222214 geno22222215 geno22222216 geno22222217 geno22222218 geno22222219 geno22222220 geno22222221 geno22222222 geno22222223 geno22222224 geno22222225 geno22222226 geno22222227 geno22222228 geno22222229 geno222222210 geno222222211 geno222222212 geno222222213 geno222222214 geno222222215 geno222222216 geno222222217 geno222222218 geno222222219 geno222222220 geno22222221 geno222222221 geno222222222 geno222222223 geno222222224 geno222222225 geno222222226 geno222222227 geno222222228 geno222222229 geno2222222210 geno2222222211 geno2222222212 geno2222222213 geno2222222214 geno2222222215 geno2222222216 geno2222222217 geno2222222218 geno2222222219 geno2222222220 geno222222221 geno2222222221 geno2222222222 geno2222222223 geno2222222224 geno2222222225 geno2222222226 geno2222222227 geno2222222228 geno2222222229 geno22222222210 geno22222222211 geno22222222212 geno22222222213 geno22222222214 geno22222222215 geno22222222216 geno22222222217 geno22222222218 geno22222222219 geno22222222220 geno2222222221 geno22222222221 geno22222222222 geno22222222223 geno22222222224 geno22222222225 geno22222222226 geno22222222227 geno22222222228 geno22222222229 geno222222222210 geno222222222211 geno222222222212 geno222222222213 geno222222222214 geno222222222215 geno222222222216 geno222222222217 geno222222222218 geno222222222219 geno222222222220 geno22222222221 geno222222222221 geno222222222222 geno222222222223 geno222222222224 geno222222222225 geno222222222226 geno222222222227 geno222222222228 geno222222222229 geno2222222222210 geno2222222222211 geno2222222222212 geno2222222222213 geno2222222222214 geno2222222222215 geno2222222222216 geno2222222222217 geno2222222222218 geno2222222222219 geno2222222222220 geno222222222221 geno2222222222221 geno2222222222222 geno2222222222223 geno2222222222224 geno2222222222225 geno2222222222226 geno2222222222227 geno2222222222228 geno2222222222229 geno22222222222210 geno22222222222211 geno22222222222212 geno22222222222213 geno22222222222214 geno22222222222215 geno22222222222216 geno22222222222217 geno22222222222218 geno22222222222219 geno22222222222220 geno2222222222221 geno22222222222221 geno22222222222222 geno22222222222223 geno22222222222224 geno22222222222225 geno22222222222226 geno22222222222227 geno22222222222228 geno22222222222229 geno222222222222210 geno222222222222211 geno222222222222212 geno222222222222213 geno222222222222214 geno222222222222215 geno222222222222216 geno222222222222217 geno222222222222218 geno222222222222219 geno222222222222220 geno22222222222221 geno222222222222221 geno222222222222222 geno222222222222223 geno222222222222224 geno222222222222225 geno222222222222226 geno222222222222227 geno222222222222228 geno222222222222229 geno2222222222222210 geno2222222222222211 geno2222222222222212 geno2222222222222213 geno2222222222222214 geno2222222222222215 geno2222222222222216 geno2222222222222217 geno2222222222222218 geno2222222222222219 geno2222222222222220 geno222222222222221 geno2222222222222221 geno2222222222222222 geno2222222222222223 geno2222222222222224 geno2222222222222225 geno2222222222222226 geno2222222222222227 geno2222222222222228 geno2222222222222229 geno22222222222222210 geno22222222222222211 geno22222222222222212 geno22222222222222213 geno22222222222222214 geno22222222222222215 geno22222222222222216 geno22222222222222217 geno22222222222222218 geno22222222222222219 geno22222222222222220 geno2222222222222221 geno22222222222222221 geno22222222222222222 geno22222222222222223 geno22222222222222224 geno22222222222222225 geno22222222222222226 geno22222222222222227 geno22222222222222228 geno22222222222222229 geno222222222222222210 geno222222222222222211 geno222222222222222212 geno222222222222222213 geno222222222222222214 geno222222222222222215 geno222222222222222216 geno222222222222222217 geno222222222222222218 geno222222222222222219 geno222222222222222220 geno22222222222222221 geno222222222222222221 geno222222222222222222 geno222222222222222223 geno222222222222222224 geno222222222222222225 geno222222222222222226 geno222222222222222227 geno222222222222222228 geno222222222222222229 geno2222222222222222210 geno2222222222222222211 geno2222222222222222212 geno2222222222222222213 geno2222222222222222214 geno2222222222222222215 geno2222222222222222216 geno2222222222222222217 geno2222222222222222218 geno2222222222222222219 geno2222222222222222220 geno222222222222222221 geno2222222222222222221 geno2222222222222222222 geno2222222222222222223 geno2222222222222222224 geno2222222222222222225 geno2222222222222222226 geno2222222222222222227 geno2222222222222222228 geno2222222222222222229 geno22222222222222222210 geno22222222222222222211 geno22222222222222222212 geno22222222222222222213 geno22222222222222222214 geno22222222222222222215 geno22222222222222222216 geno22222222222222222217 geno22222222222222222218 geno22222222222222222219 geno22222222222222222220 geno2222222222222222221 geno22222222222222222221 geno22222222222222222222 geno22222222222222222223 geno22222222222222222224 geno22222222222222222225 geno22222222222222222226 geno22222222222222222227 geno22222222222222222228 geno22222222222222222229 geno222222222222222222210 geno222222222222222222211 geno222222222222222222212 geno222222222222222222213 geno222222222222222222214 geno222222222222222222215 geno222222222222222222216 geno222222222222222222217 geno222222222222222222218 geno222222222222222222219 geno222222222222222222220 geno22222222222222222221 geno222222222222222222221 geno222222222222222222222 geno222222222222222222223 geno222222222222222222224 geno222222222222222222225 geno222222222222222222226 geno222222222222222222227 geno222222222222222222228 geno222222222222222222229 geno2222222222222222222210 geno2222222222222222222211 geno2222222222222222222212 geno2222222222222222222213 geno2222222222222222222214 geno2222222222222222222215 geno2222222222222222222216 geno2222222222222222222217 geno2222222222222222222218 geno2222222222222222222219 geno2222222222222222222220 geno222222222222222222221 geno2222222222222222222221 geno2222222222222222222222 geno2222222222222222222223 geno2222222222222222222224 geno2222222222222222222225 geno2222222222222222222226 geno2222222222222222222227 geno2222222222222222222228 geno2222222222222222222229 geno22222222222222222222210 geno22222222222222222222211 geno22222222222222222222212 geno22222222222222222222213 geno22222222222222222222214 geno22222222222222222222215 geno22222222222222222222216 geno22222222222222222222217 geno22222222222222222222218 geno22222222222222222222219 geno22222222222222222222220 geno2222222222222222222221 geno22222222222222222222221 geno22222222222222222222222 geno22222222222222222222223 geno22222222222222222222224 geno22222222222222222222225 geno22222222222222222222226 geno22222222222222222222227 geno22222222222222222222228 geno22222222222222222222229 geno222222222222222222222210 geno222222222222222222222211 geno222222222222222222222212 geno222222222222222222222213 geno222222222222222222222214 geno222222222222222222222215 geno222222222222222222222216 geno222222222222222222222217 geno222222222222222222222218 geno222222222222222222222219 geno222222222222222222222220 geno22222222222222222222221 geno222222222222222222222221 geno222222222222222222222222 geno222222222222222222222223 geno222222222222222222222224 geno222222222222222222222225 geno222222222222222222222226 geno222222222222222222222227 geno222222222222222222222228 geno222222222222222222222229 geno2222222222222222222222210 geno2222222222222222222222211 geno2222222222222222222222212 geno2222222222222222222222213 geno2222222222222222222222214 geno2222222222222222222222215 geno2222222222222222222222216 geno2222222222222222222222217 geno2222222222222222222222218 geno2222222222222222222222219 geno2222222222222222222222220 geno222222222222222222222221 geno222222222222222222222221 geno2222222222222222222222222 geno2222222222222222222222223 geno2222222222222222222222224 geno2222222222222222222222225 geno2222222222222222222222226 geno2222222222222222222222227 geno2222222222222222222222228 geno2222222222222222222222229 geno22222222222222222222222210 geno22222222222222222222222211 geno22222222222222222222222212 geno22222222222222222222222213 geno22222222222222222222222214 geno22222222222222222222222215 geno22222222222222222222222216 geno22222222222222222222222217 geno22222222222222222222222218 geno22222222222222222222222219 geno22222222222222222222222220 geno2222222222222222222222221 geno2222222222222222222222221 geno22222222222222222222222222 geno2222222222222222222222223 geno2222222222222222222222224 geno2222222222222222222222225 geno2222222222222222222222226 geno2222222222222222222222227 geno2222222222222222222222228 geno2222222222222222222222229 geno22222222222222222222222210 geno22222222222222222222222211 geno22222222222222222222222212 geno22222222222222222222222213 geno22222222222222222222222214 geno22222222222222222222222215 geno22222222222222222222222216 geno22222222222222222222222217 geno22222222222222222222222218 geno22222222222222222222222219 geno22222222222222222222222220 geno2222222222222222222222221 geno2222222222222222222222221 geno22222222222222222222222222 geno22222222222222222222222223 geno22222222222222222222222224 geno22222222222222222222222225 geno22222222222222222222222226 geno22222222222222222222222227 geno22222222222222222222222228 geno22222222222222222222222229 geno222222222222222222222222210 geno222222222222222222222222211 geno222222222222222222222222212 geno222222222222222222222222213 geno222222222222222222222222214 geno222222222222222222222222215 geno222222222222222222222222216 geno222222222222222222222222217 geno222222222222222222222222218 geno222222222222222222222222219 geno222222222222222222222222220 geno22222222222222222222222221 geno22222222222222222222222221 geno222222222222222222222222222 geno22222222222222222222222223 geno22222222222222222222222224 geno22222222222222222222222225 geno
```

## Modelo de interação – Parte 2

Na primeira parte cuidamos dos dados de entrada, agora será feito os dados de saída para isso no código fonte serão necessárias algumas alterações.

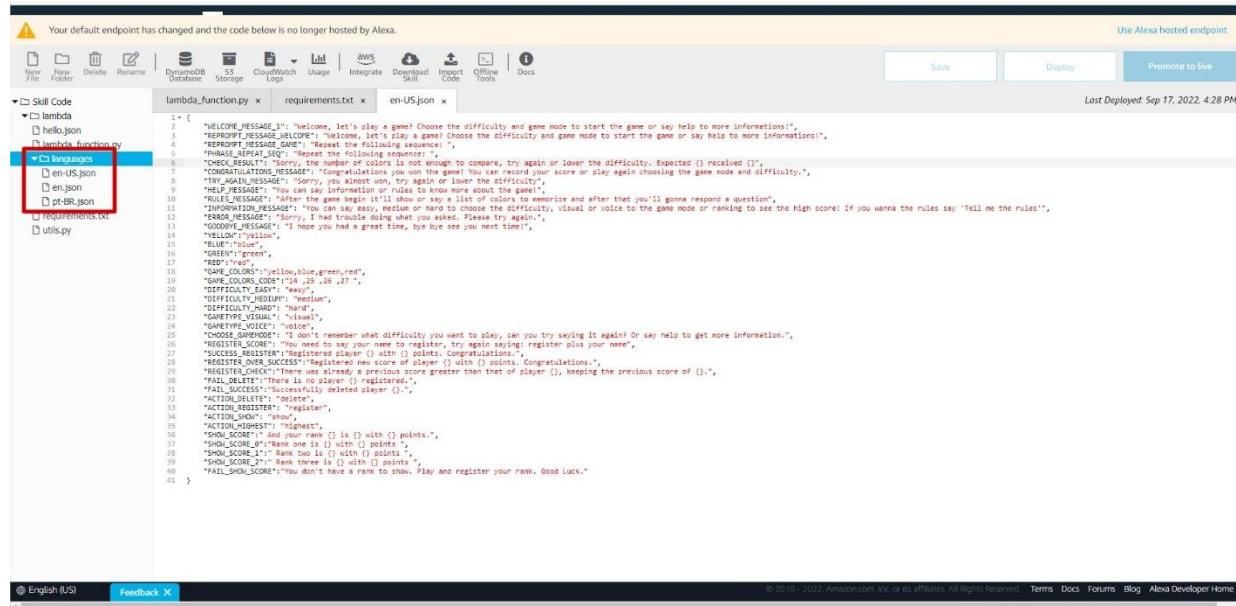
Teremos que utilizar o conceito de interceptor, existem duas variantes de interceptores o de resposta que será executado após finalizar algum método e o de requisição que será executado antes de entrar no método. No caso do idioma é necessário utilizar o de requisição.

Para isso, devemos importar o componente "AbstractRequestInterceptor". Abaixo se encontra a linha padrão para importar os interceptores mais utilizados.

```
from ask_sdk_core.dispatch_components import (AbstractRequestHandler,
AbstractExceptionHandler,AbstractResponseInterceptor,AbstractRequestInterceptor)
```

Em seguida, para ficar legível e de fácil controle o ideal é criar uma pasta com documentos JSON que irá tratar das strings de cada idioma. Você pode criar qualquer estrutura de JSON, neste caso deixamos simples apenas com um id para cada valor. Conforme figura 70.

Figura 70 - Idiomas JSON



```
1+ {
  "WELCOME_MESSAGE": "Hello, let's play! Choose the difficulty and game mode to start the game or say help to more informations!",
  "REPROMPT_MESSAGE_WELCOME": "Welcome, let's play a game! Choose the difficulty and game mode to start the game or say help to more informations!",
  "REPROMPT_MESSAGE_GAME": "Repeat the following sequence: ",
  "HELP_MESSAGE": "You can say informations or rules to know more about the game!",
  "CHECK_RESULT": "Sorry, the color you chose is not enough to compare, try again or lower the difficulty. Expected [1] received [2]",
  "CONGRATULATIONS_MESSAGE": "Congratulations, you won the game! You can record your score or play again choosing the game mode and difficulty.",
  "HIGH_SCORE_MESSAGE": "Your current high score is [1]. Try to beat it next time!",
  "RULE_MESSAGE": "After the game begin it'll show or say a list of colors to memorize and after that you'll gonna respond a question",
  "RULES_MESSAGE": "Tell me the rules of the game, I will tell you what difficulty, visual or voice to the game mode or ranking to see the high score! If you wanna the rules say 'Tell me the rules'",
  "ERROR_MESSAGE": "Sorry, I had trouble doing what you asked. Please try again!",
  "GOODBYE_MESSAGE": "I hope you had a great time, bye bye see you next time!",
  "COLOR_MESSAGE": "The color is [1]. Try to guess it!",
  "COLOR_HELP_MESSAGE": "The color is [1]. Try to guess it!",
  "COLOR_WRONG_MESSAGE": "The color is [1]. Try to guess it!",
  "COLOR_CORRECT_MESSAGE": "The color is [1]. You're right!",
  "COLOR_HELP_REPEAT_MESSAGE": "The color is [1]. Try to guess it!",
  "COLOR_WRONG_REPEAT_MESSAGE": "The color is [1]. Try to guess it!",
  "COLOR_CORRECT_REPEAT_MESSAGE": "The color is [1]. You're right!",
  "GAME_COLORS": "yellow,blue,green,red",
  "DIFFICULTY_EASY": "easy",
  "DIFFICULTY_MEDIUM": "medium",
  "DIFFICULTY_HARD": "hard",
  "GAME_TYPE_VOICE": "voice",
  "GAME_TYPE_VISUAL": "visual",
  "GAME_TYPE_TEXT": "text",
  "REGISTER_SCORE": "You need to say your name to register, try again saying register plus your name",
  "SUCCESS_REGISTER": "Registered player [1] with [2] points. Congratulations!",
  "REGISTER_CHECK": "There was already a previous score greater than that of player [1], keeping the previous score of [2].",
  "FAIL_DELETE": "There is no player [1] registered.",
  "FAIL_REGISTER": "Player [1] already deleted player [2].",
  "ACTION_DELETE": "Delete",
  "ACTION_REGISTER": "Register",
  "ACTION_HIGHSCORE": "highest",
  "SHOW_SCORE_1": "Rank one is [1] with [2] points.",
  "SHOW_SCORE_0": "Rank one is [1] with [2] points",
  "SHOW_SCORE_2": "Rank two is [1] with [2] points",
  "SHOW_SCORE_3": "Rank three is [1] with [2] points",
  "FAIL_SHOW_SCORE": "You don't have a rank to show. Play and register your new. Good Luck."
}
```

Agora devemos criar um método passando como parâmetro o tipo do interceptor, nesse método basicamente será identificar no JSON de entrada qual é a localidade recebida e abrir o arquivo JSON e passar os dados para a sessão.

```
class LocalizationInterceptor(AbstractRequestInterceptor):

    def process(self, handler_input):
        locale = handler_input.request_envelope.request.locale
        logger.info("Locale is {}".format(locale))
        try:
            with open("languages/" + str(locale) + ".json") as language_data:
                language_prompts = json.load(language_data)
        except:
            with open("languages/" + str(locale[:2]) + ".json") as language_data:
```

```
language_prompts = json.load(language_data)
handler_input.attributes_manager.request_attributes["_"] = language_prompts
```

Por fim, nos demais métodos seria apenas buscar na sessão o valor do id necessário. No quadro a seguir é possível verificar um exemplo simples:

```
class InitializeGameIntentHandler(AbstractRequestHandler):
    """Handler for Initialize Game Intent."""
    def can_handle(self, handler_input):
        # type: (HandlerInput) -> bool
        return ask_utils.is_intent_name("InitializeGameIntent")(handler_input)

    def handle(self, handler_input):
        # type: (HandlerInput) -> Response
        language_prompts = handler_input.attributes_manager.request_attributes["_"] #Buscar o valor
        da sessão
        speak_output = language_prompts["PHRASE_REPEAT_SEQ"] #Buscar pela tag qual é a frase

        return (
            handler_input.response_builder
            .speak(speak_output)
            .set_should_end_session(False)
            .response
        )
```

Ressalto que é necessário colocar no construtor este método criado.

```
sb.add_global_request_interceptor(LocalizationInterceptor())
```

## Dados persistentes em servidor hospedado Alexa

Diferente do que foi feito nas sessões para se utilizar o DynamoDB na skill, caso o servidor da skill seja hospedado na própria Alexa devemos utilizar os próprios componentes da ask-sdk disponibilizados pela Amazon, visto que o usuário dos serviços Lambdas é gerado de forma aleatória e não temos controle das regras de acesso dele e por isso não conseguimos realizar os mesmos passos descritos no item “DynamoDB”. Por isso o ideal é utilizar o serviço S3 bucket por ser simples a sua utilização, pelo código ser curto e os dados a serem salvados pequenos a utilização dele é o suficiente .

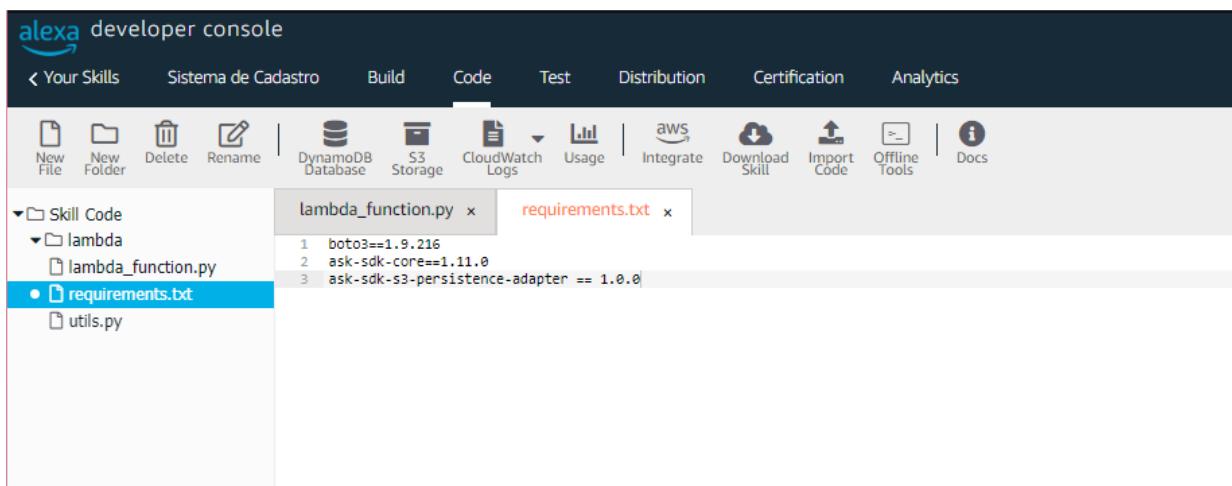
### Alterações no código

No arquivo de “requirements.txt” da skill, deve-se inserir a seguinte linha:

```
ask-sdk-dynamodb-persistence-adapter==1.15.0
```

Na figura 71 é possível validar como irá ficar após a inserção desta linha.

Figura 711 - Arquivo 'requirements.txt'



Abra o arquivo 'lambda\_function.py', e insira as linhas que não existir:

```
import os
import ask_sdk_core.utils as ask_utils
from ask_sdk_s3.adapter import S3Adapter
s3_adapter = S3Adapter(bucket_name=os.environ["S3_PERSISTENCE_BUCKET"])
```

Será necessário substituir a linha:

```
from ask_sdk_core.skill_builder import SkillBuilder
```

Pela linha:

```
from ask_sdk_core.skill_builder import CustomSkillBuilder
```

Exemplo de como poderá ficar após essas alterações:

```
import logging
import ask_sdk_core.utils as ask_utils
import os
from ask_sdk_s3.adapter import S3Adapter
s3_adapter = S3Adapter(bucket_name=os.environ["S3_PERSISTENCE_BUCKET"])

from ask_sdk_core.skill_builder import CustomSkillBuilder
from ask_sdk_core.dispatch_components import AbstractRequestHandler
from ask_sdk_core.dispatch_components import AbstractExceptionHandler
from ask_sdk_core.handler_input import HandlerInput
from ask_sdk_model import Response

logger = logging.getLogger(name)
logger.setLevel(logging.INFO)
```

Na parte inferior do arquivo também será necessário alterar outra linha de código, no caso deverá procurar essa linha:

```
sb = SkillBuilder()
```

E substitui-la por esta linha:

```
sb = CustomSkillBuilder(persistence_adapter=s3_adapter)
```

Após esses passos utilizar os botões “Save” e “Deploy” para testar se a skill irá abrir sem erros.

### Salvando dados

Para salvar os dados é apenas necessário utilizar duas linhas:

```
handler_input.attributes_manager.persistent_attributes = banco  
handler_input.attributes_manager.save_persistent_attributes()
```

Sendo o que o valor “banco” são os dados que precisa guardar.

### Recuperando os dados

Para pegar os dados salvos é preciso apenas utilizar a linha:

```
banco = handler_input.attributes_manager.persistent_attributes
```

Sendo que a variável “banco” irá receber todas as informações que já estão guardadas, com isso é possível manipular os dados e posteriormente utilizar o comando de salvar novamente.

## Fóruns

A Amazon conta com diversos fóruns específicos para retirar dúvidas de como criar skills ou de recursos específicos dos serviços, sendo possível conversar com os desenvolvedores da Alexa. Será lista os principais fóruns e o slack da comunidade.

Fórum para postar ideias do que pode ser desenvolvido para a Alexa:

<https://alexa.design/uservoice>

Fórum voltado para serviços da AWS:

<https://repost.aws>

Fórum focado para retirar dúvidas de funcionamento ou como criar algum recurso da skill:

<https://amazon.developer.forums.answerhub.com/index.html>

Slack da comunidade que conta com os programadores da Alexa além de outros programadores:

<https://alexacomunity.slack.com>

## Referências

<https://developer.amazon.com/en-US/docs/alexa/custom-skills/choose-the-invocation-name-for-a-custom-skill.html>

<https://apl.ninja>

<https://developer.amazon.com/en-US/docs/alexa/alexa-presentation-language/apl-commands.html>

<https://dabblelab.com/tutorials>

<https://apl.ninja/xeladotbe/you-win-0mfh>

<https://apl.ninja/document/xeladotbe/jeff-blankenburgs-twitch-stream-intro-k8a9>

<https://aws.amazon.com/iot-core/>

<https://aws.amazon.com/cloudwatch/features/#:~:text=CloudWatch%20enables%20you%20to%20monitor,building%20applications%20and%20business%20value.>

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html>

<https://developer.amazon.com/en-US/docs/alexa/custom-skills/steps-to-build-a-custom-skill.html>

<https://www.amebaito.com/en/ameba-arduino-amazon-alexa/>

[https://aws.amazon.com/pt/lambda/?nc2=h\\_m1](https://aws.amazon.com/pt/lambda/?nc2=h_m1)

<https://developer.amazon.com/en-US/docs/alexa/ask-overviews/what-is-the-alexa-skills-kit.html>

<https://github.com/alexa-samples/skill-sample-python-first-skill/tree/master/module-4>

<https://medium.com/captech-corner/automating-a-tower-fan-with-alexa-skills-aws-lambda-aws-iot-and-an-esp32-ad0d4ba1da22>

<https://developer.amazon.com/en-US/docs/alexa/custom-skills/choose-the-invocation-name-for-a-custom-skill.html>

<https://developer.amazon.com/en-US/docs/alexa/custom-skills/create-intents-utterances-and-slots.html>

<https://www.youtube.com/watch?v=8zhv6GDSDE8>

<https://www.amebaito.com.cn/en/amebad-arduino-aws-shadow/>

<https://developer.amazon.com/en-US/docs/alexa/interaction-model-design/internationalize-the-interaction-model-for-your-skill.html>

<https://www.alexanetcore.com/internationalization/>