

## 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 X – Passo 1 -> Configurando Arduino para o ESP32

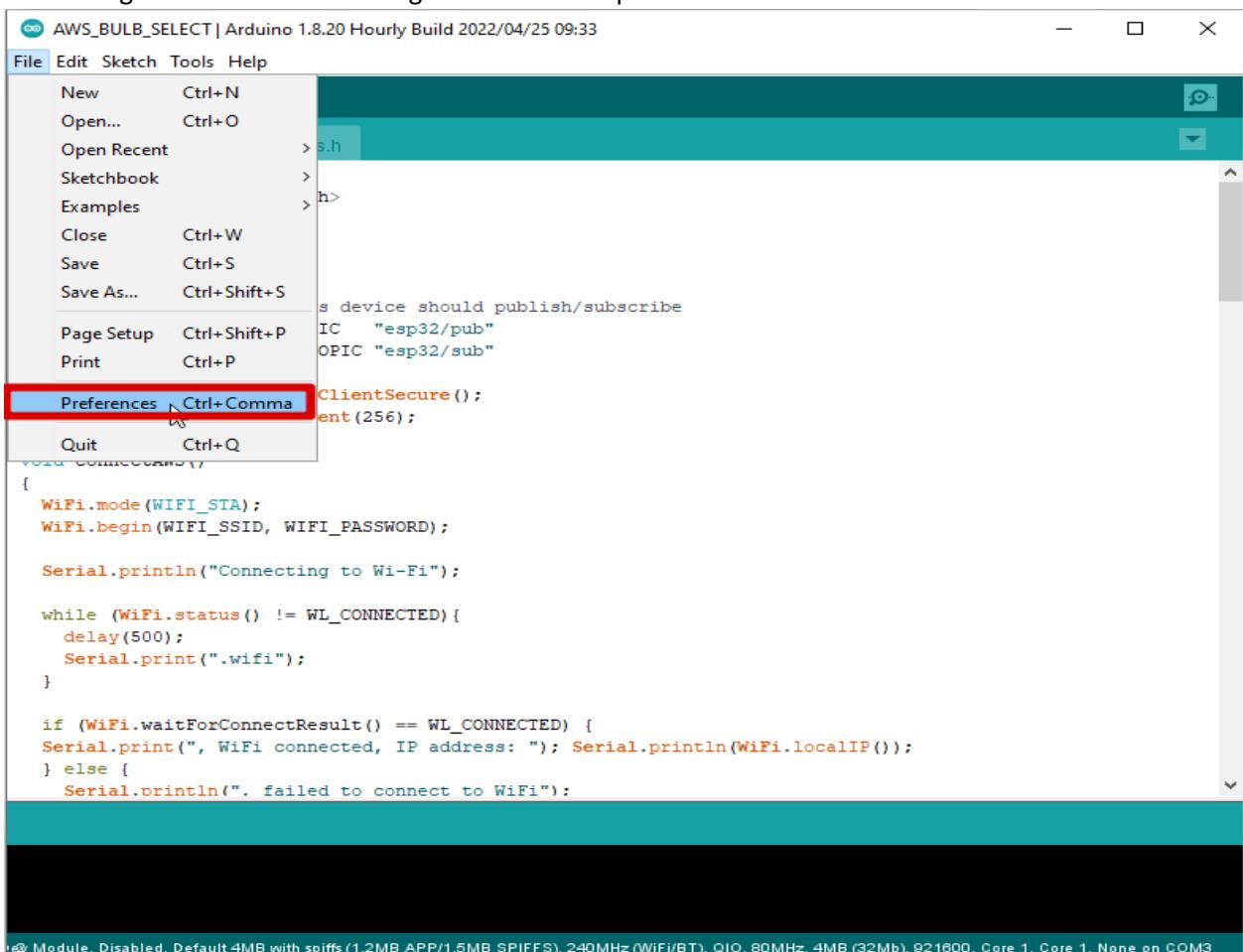


Figura X – Passo 2 -> Configurando Arduino para o ESP32 (continuação)

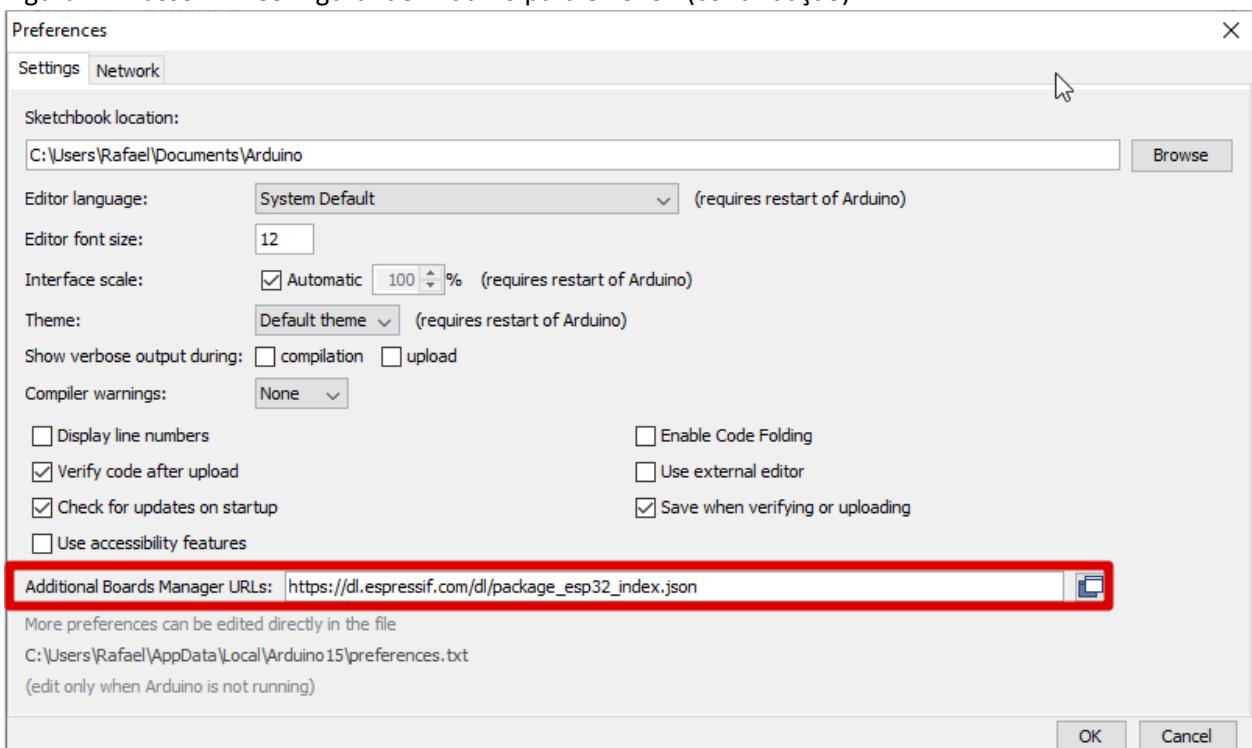


Figura X – Passo 3 -> Instalando pacote ESP32

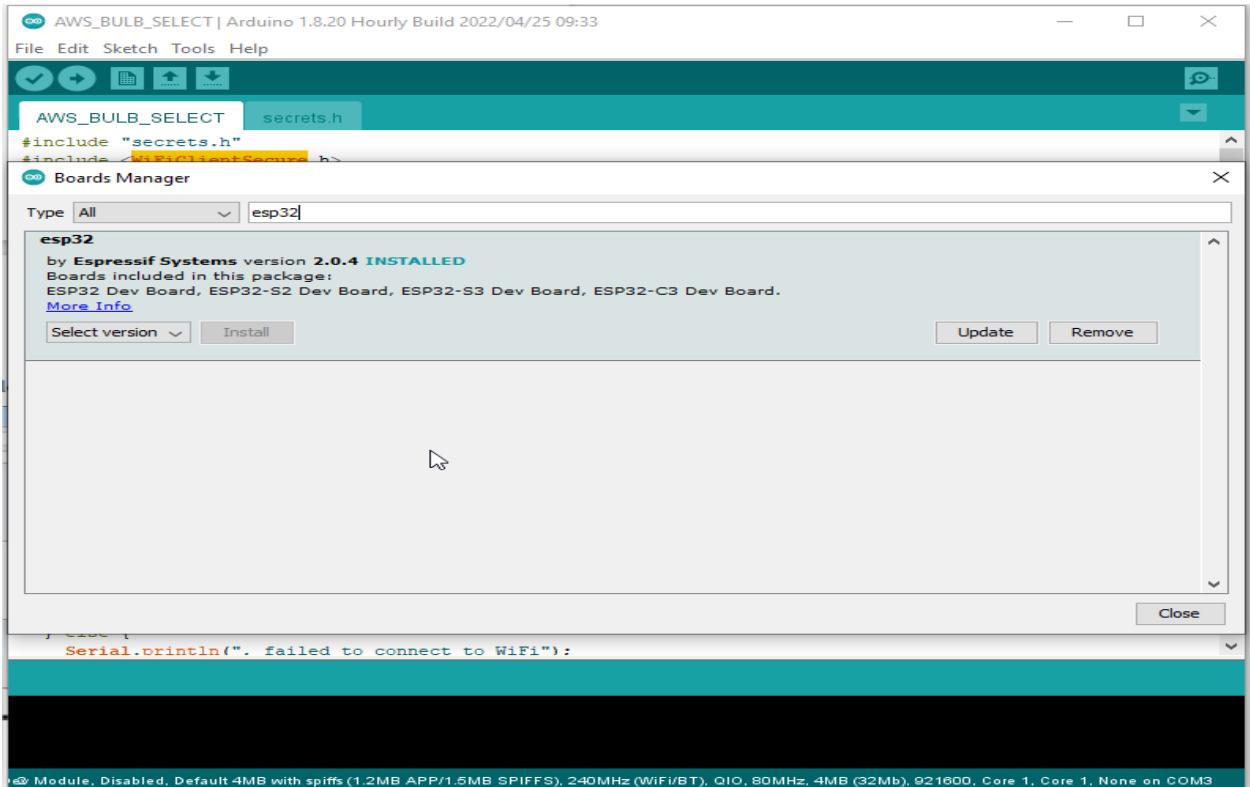


Figura X – Passo 4 -> Selecionando placa ESP32

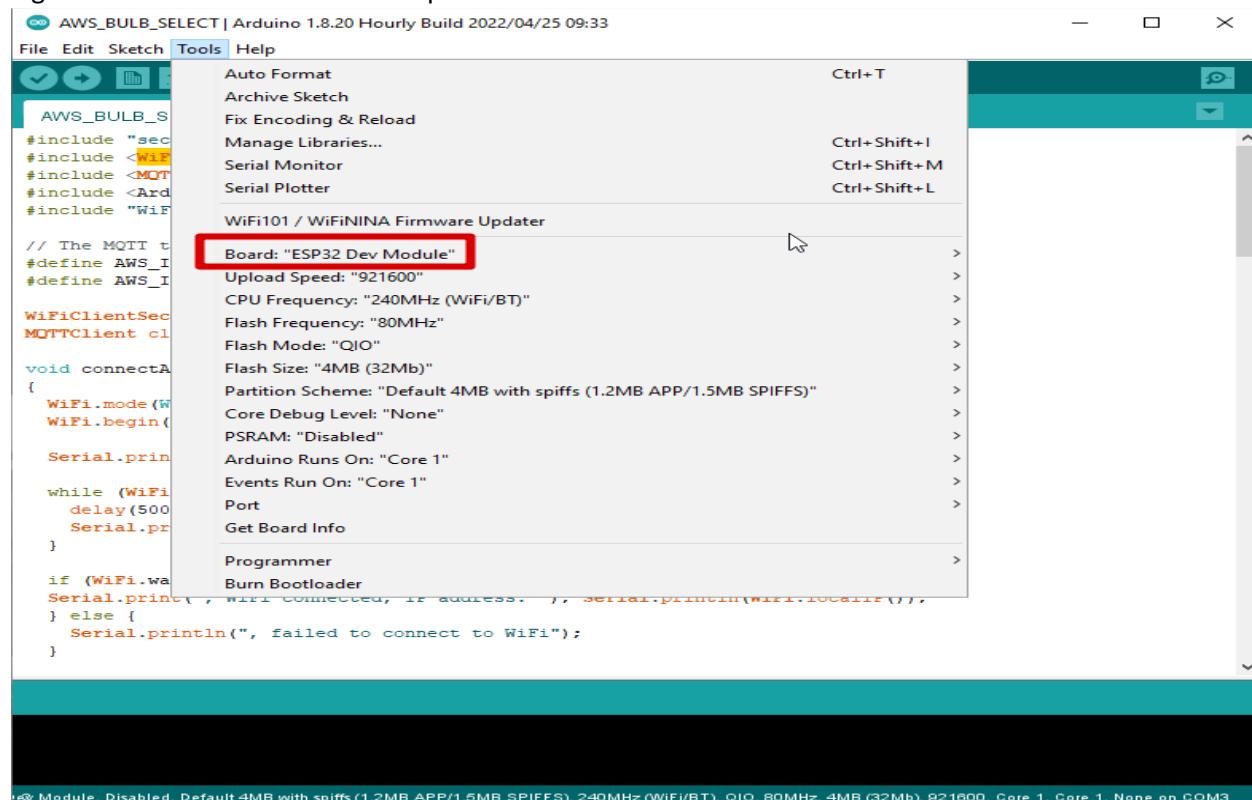


Figura X – Passo 5 -> Instalando biblioteca MQTT

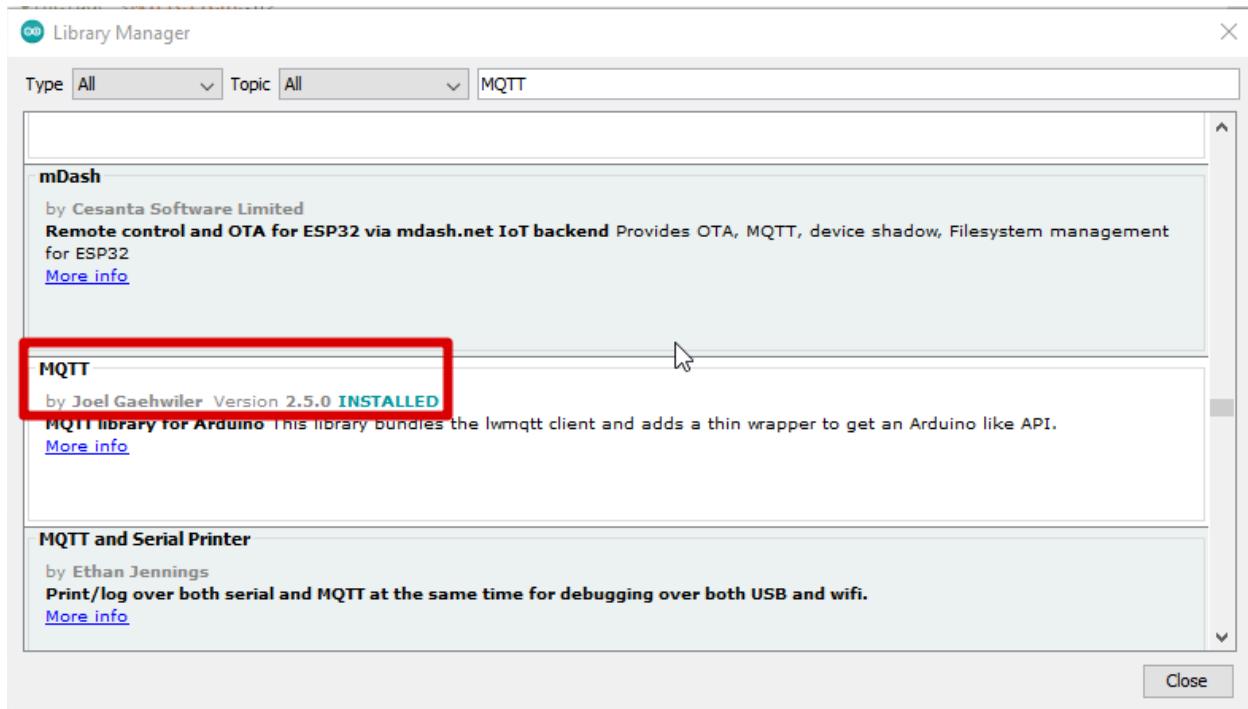
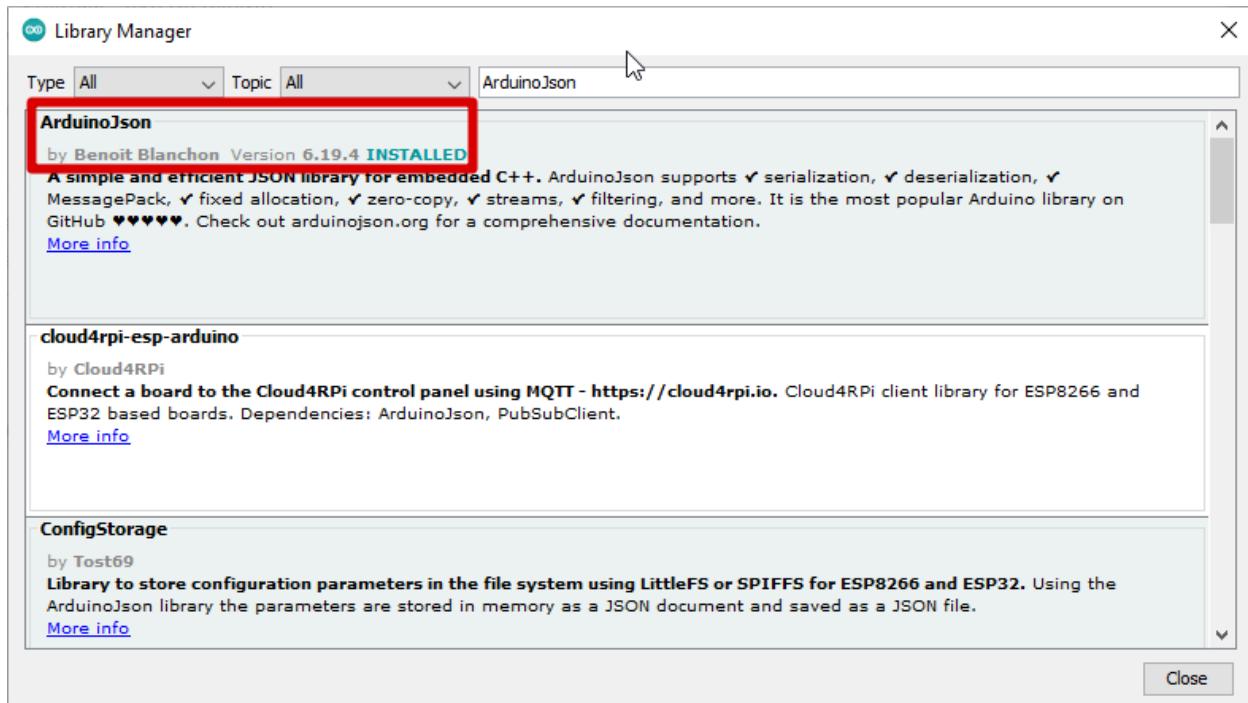


Figura X – Passo 6 -> Instalando biblioteca ArduinoJson



## Código Arduino IDE

O código a seguir foi disponibilizado no github pelo link: xxx

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);
}
```



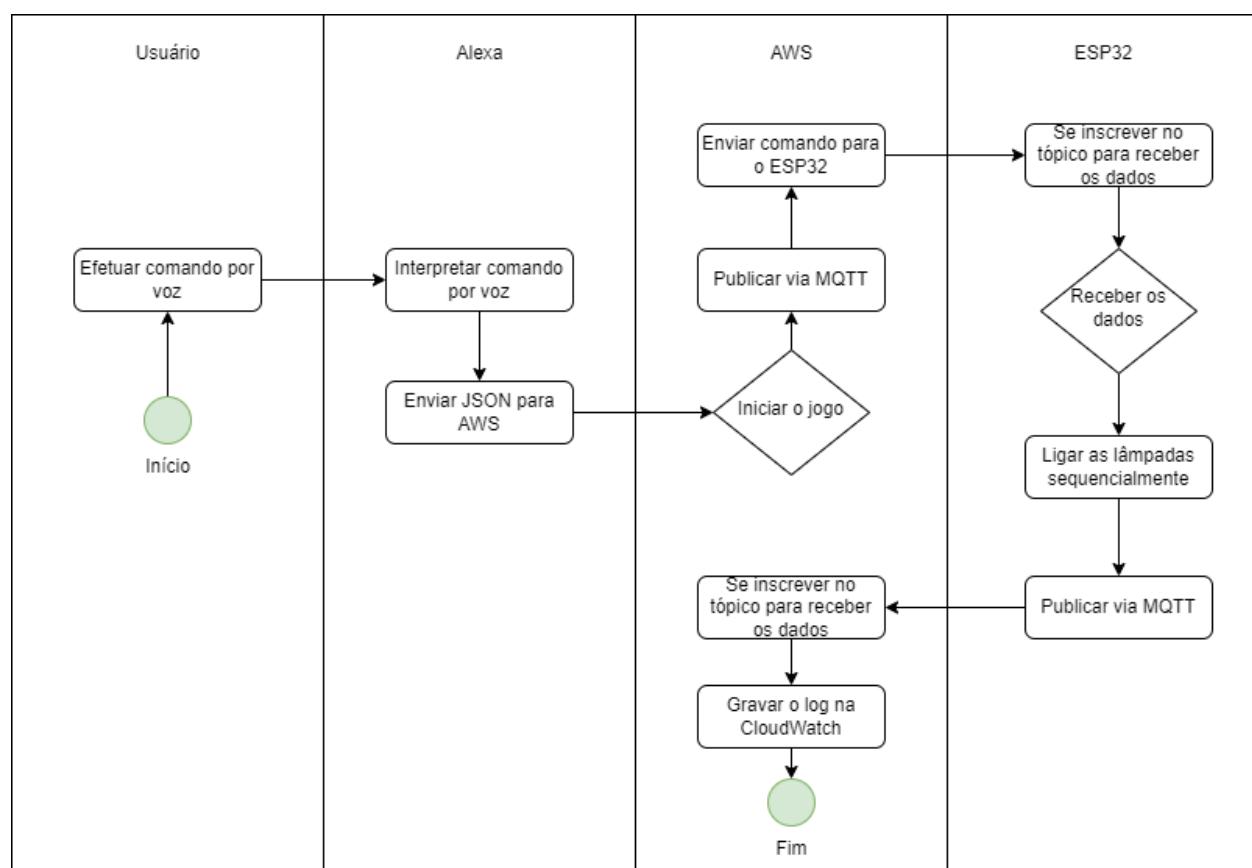
## Criação de uma skill para a Amazon Alexa

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 interativos, 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>

### 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 tomar a ação de acordo com o comando. Se for um comando para alterar o estado da luz ou abrir a porta, o AWS envia a requisição para o Arduino efetuar a ação correspondente.



### Conceitos

Neste item será apresentado como realizar a criação de uma skill customizável 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.

**Traduzir e colocar a imagem como citação direta?**

<https://developer.amazon.com/en-US/docs/alexa/workshops/build-an-engaging-skill/why-build/index.html>

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.

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

**Intenções**

Uma intenção 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 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.

<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>

## Console do desenvolvedor Alexa

A partir da Figura X até X é mostrado como configurar uma nova Skill. Será utilizado a localização do Brasil, mas existem algumas funcionalidades disponíveis apenas em outras localidades.

Figura X – Página inicial do site da Amazon Alexa

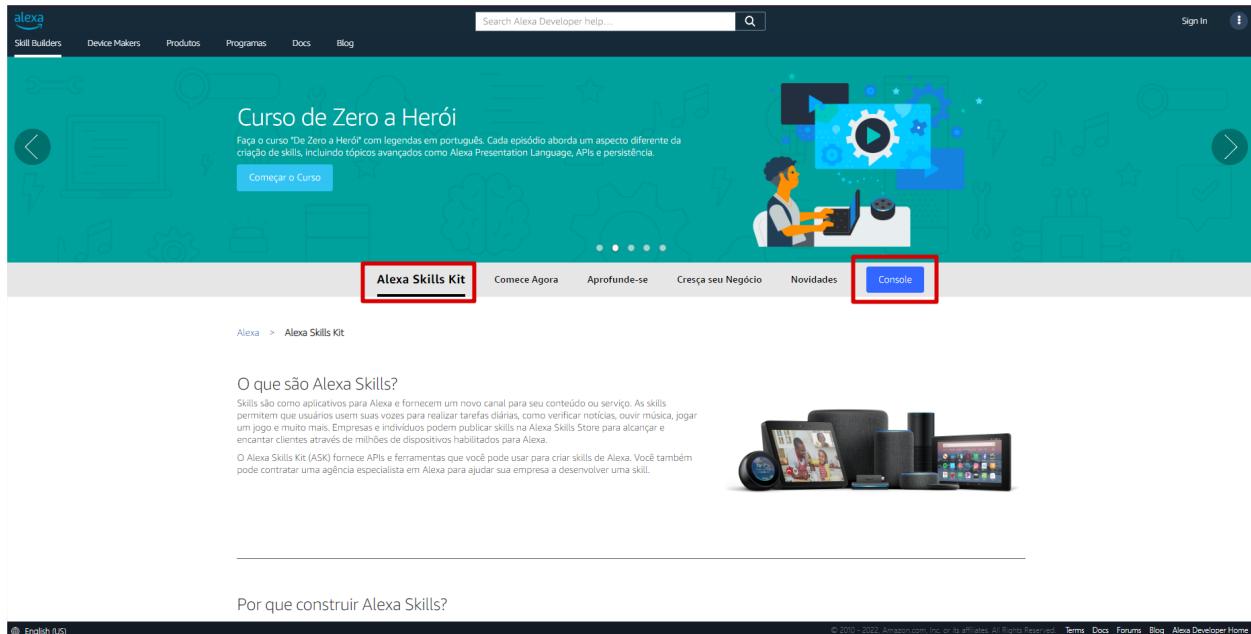


Figura X – Página de login para o console do desenvolvedor

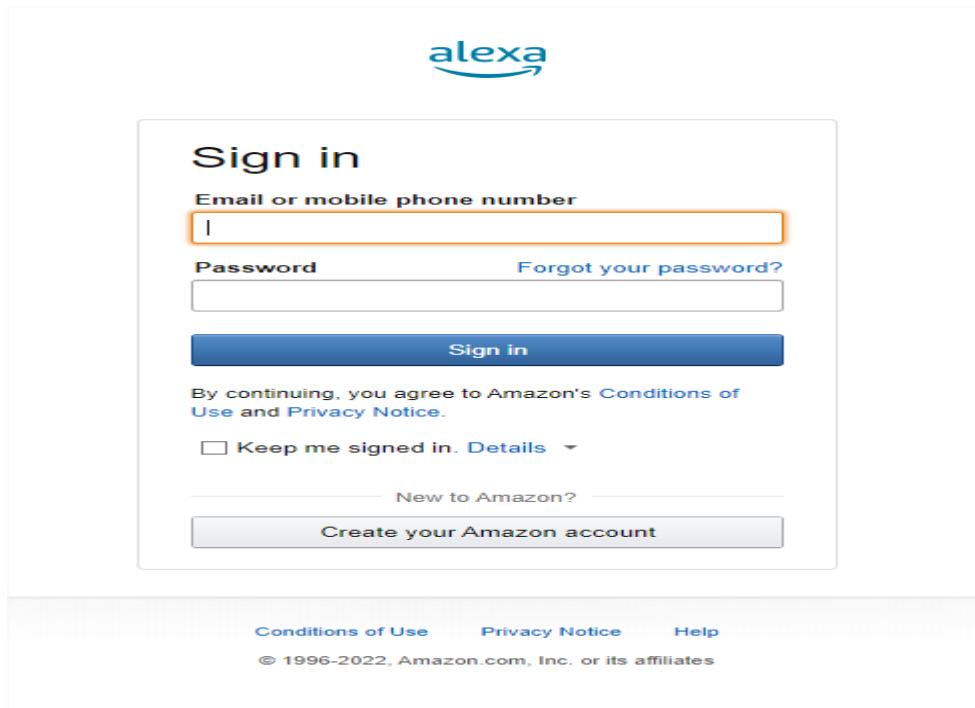


Figura X – Página para gerenciamos das skills

livro\_miguel has successfully been deleted.

**Skills** Earnings Payments Hosting Settings

Alexa Skills Skill examples Learn more

Search by skill name or skill ID

Create Skill

SKILL NAME	LANGUAGE	MODIFIED	STATUS	ACTIONS
TCC V1N Custom-Copy Skill ID	Portuguese (BR) +1	2022-09-25	● In Dev	Choose action
TEST_ESP3 Custom-Copy Skill ID	English (US)	2022-09-21	● In Dev	Choose action
TCC V1 Custom-Copy Skill ID	Portuguese (BR) +1	2022-08-28	● In Dev	Choose action
Raízes de uma equação de 2 grau Custom-Copy Skill ID	Portuguese (BR)	2022-08-13	● In Dev	Choose action
Sistema de Cadastro Custom-Copy Skill ID	Portuguese (BR)	2022-08-13	● In Dev	Choose action

Feedback X

Alexa Skill Insights

To Dos

Check back for more

The insights in this section have been updated to show opportunities to improve your skills. We'll have suggestions for your skills soon.

© 2010 - 2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms Docs Forums Blog Alexa Developer Home

Figura X – Selecionar o idioma e modelo Custom

Create a new skill

Skill name: TCC V1N

Primary locale: Portuguese (BR)

Choose a model to add to your skill

1. Choose a model to add to your skill

2. Choose a method to host your skill's backend resources

Model: Custom  
Host: Alexa-hosted (Node.js)  
Hosting Region: US East (N. Virginia)

Custom (Selected)  
Design a unique experience for your users. A custom model enables you to create all of your skill's interactions.

Flash Briefing  
Give users control of their news feed. This pre-built model lets users control what updates they listen to.

Smart Home  
Give users control of their smart home devices. This pre-built model lets users turn off the lights and other devices without getting up.

Video  
Let users find and consume video content. This pre-built model supports content searches and content suggestions.

"Alexa, acenda as luzes da cozinha"  
"Alexa, reproduzo Interstellar"

Alexa, quais são as notícias?

Feedback X

© 2010 - 2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms Docs Forums Blog Alexa Developer Home

Figura X – Selecionar linguagem de programação

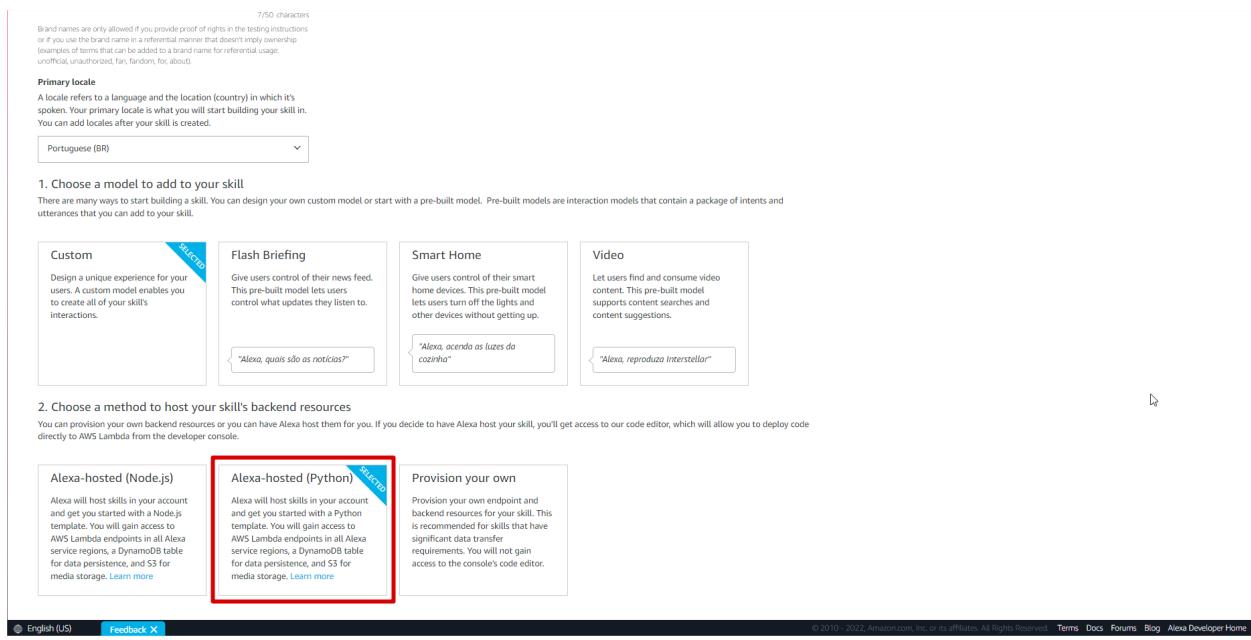


Figura X – Selecionar “Start from Scratch”

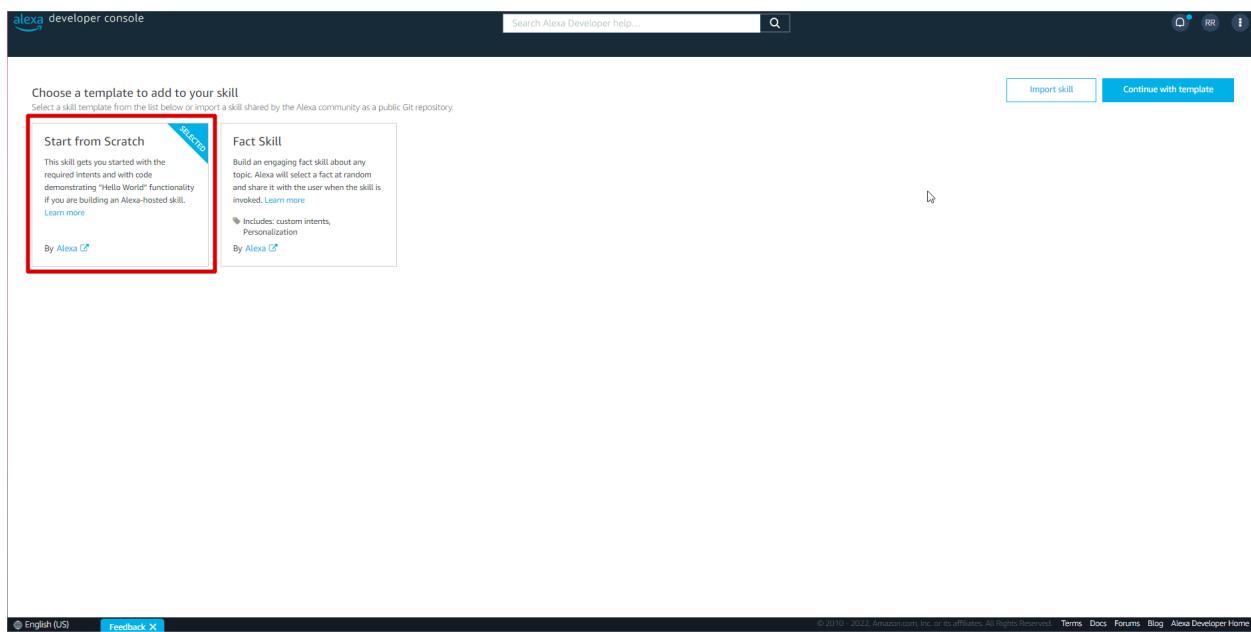


Figura X – Inserindo um nome de invocação da skill

The screenshot shows the Alexa developer console interface. The left sidebar has sections like 'Your Skills', 'TCC PTBR', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. Under 'CUSTOM', there's a 'Skill Invocation Name' section with 'Skill Launch Phrases' and 'Intent Launch Phrases'. Below that is 'Interaction Model', 'Assets', and 'Slot Types (0)'. Under 'MODELS', there's 'Multimodal Responses', 'Interfaces', and 'Endpoint'. Under 'TOOLS', there's 'Errors and Warnings' and 'Feedback'. The main content area is titled 'Invocation' with a note about skill invocation names. It shows a sample user interaction: 'User: Alexa, ask daily horoscopes for the horoscope for Gemini'. Below it, a red box highlights the 'Skill Invocation Name' input field which contains the value 'algoritmo trinta'. There's also a note about brand names and a link to 'How to pick names that are right for you'.

Figura X – Adicionando novo intent

The screenshot shows the Alexa developer console interface. The left sidebar lists various intent categories and their sub-items, such as 'Skill Invocation Name', 'Skill Launch Phrases', 'Intent Launch Phrases', 'Interaction Model', 'Intents (11)', and 'Built-In Intents (4)'. Under 'Intents (11)', there are items like 'InitializeGameIntent', 'ScoreIntent', 'InformationIntent', 'RankingIntent', 'ShowRankingIntent', and 'ScoreNumberIntent'. A red box highlights the '+ Add Intent' button. The main content area is titled 'Intents' and shows a table of existing intents. The table columns are NAME, UTTERANCES, SLOTS, TYPE, and ACTIONS. The rows include 'AMAZON.CancelIntent', 'AMAZON.HelpIntent', 'AMAZON.StopIntent', 'AMAZON.NavigateHomeIntent', 'InitializeGameIntent' (with 3 utterances and 2 slots), 'ScoreIntent' (with 3 utterances and 1 slot), 'InformationIntent' (with 3 utterances and 1 slot), 'RankingIntent' (with 6 utterances and 3 slots), and 'ShowRankingIntent' (with 1 utterance and 2 slots). Each row has an 'Edit' or 'Edit | Delete' link under the ACTIONS column.

Figura X – Criando um intent personalizável

The screenshot shows the Alexa Developer Console interface. On the left, the navigation pane includes 'Your Skills', 'TCC PBR', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. The 'Build' tab is active. The main area displays the 'Interaction Model' for a skill. Under 'Intents (11)', the 'ShowRankingIntent' intent is currently selected. The 'Intent Slots (2)' section lists a slot named 'playerName' with a 'SLOT TYPE' dropdown set to 'AMAZON.FirstName'. There are also sections for 'Sample Utterances (1)', 'Dialog Delegation Strategy', and 'Intent Slots (2)'. The bottom of the screen shows standard developer tools like 'Feedback', 'Errors and Warnings', and links to 'Terms', 'Docs', 'Forums', 'Blog', and 'Alexa Developer Home'.

Figura X – Criando um intent e slot personalizável

This screenshot shows the 'ScoreIntent' intent selected in the left sidebar. In the 'Intent Slots (1)' section, a slot named 'SequenceColors' is listed with its 'SLOT TYPE' set to 'Colors', which is highlighted with a red box. Below this, there is a row for 'Create a new slot' with a 'Select a slot type' dropdown. The rest of the interface is similar to the previous screenshot, including sections for 'Sample Utterances (3)', 'Dialog Delegation Strategy', and 'Intent Slots (1)'. The bottom of the screen includes 'Feedback', 'Errors and Warnings', and developer links.

Figura X – Adicionando valores no slot personalizado

Aqui você deve colocar qual a declaração para ativar essa intenção.

Figura X – inserindo uma declaração

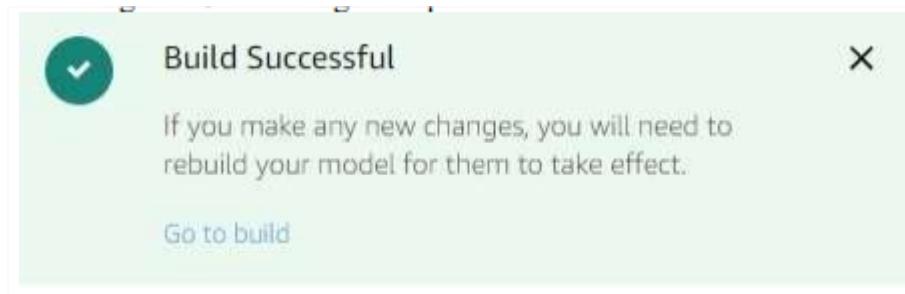
Neste exemplo essa declaração serve apenas para iniciar o jogo

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

Figura X – inserindo uma declaração

The screenshot shows the Alexa developer console interface. The top navigation bar includes 'Your Skills', 'TCC PTBR', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. A search bar is at the top right. The left sidebar lists skill components: 'CUSTOM', 'Invocations', 'Interaction Model', 'Intents (11)', and 'InitializeGameIntent' (which is selected and expanded). Under 'InitializeGameIntent', categories like 'difficulty', 'gametype', 'ScoreIntent', etc., are listed. The main content area shows 'Sample Utterances (4)'. One example is 'iniciar o jogo no modo [difficulty] e [gametype]'. Below this is 'Dialog Delegation Strategy' set to 'fallback to skill setting'. The 'Intent Slots (2)' section shows 'name' and 'value' slots. At the bottom, there are links for 'Feedback', 'Errors and Warnings', and 'English (US)'. The footer includes copyright information and links to 'Terms', 'Docs', 'Forums', 'Blog', and 'Alexa Developer Home'.

Figura X – Mensagem de sucesso



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

Figura X – Importando JSON

The screenshot shows the Alexa developer console with the 'JSON Editor' tab selected. The left sidebar lists various skill components: 'Built-in Intents (4)', 'Annotation Sets', 'Utterance Conflicts (0)', 'Assets', 'Slot Types (7)', 'Action', 'AMAZON.FirstName', 'AMAZON.NUMBER', 'Colors', 'Difficulty', 'GameTypes', 'Rules', 'Multimodal Responses', 'Interfaces', 'Endpoint', 'MODELS', and 'TOOLS'. The main area is titled 'JSON Editor' and contains a code editor with the JSON for the 'InitializeGameIntent'. The JSON code is as follows:

```

1 "interactionModel": {
2   "languageModel": {
3     "name": "algoritmo vinte",
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": "InitializeGameIntent",
23      "slots": [
24        {
25          "name": "difficulty",
26          "type": "difficulty",
27          "samples": [
28            {
29              "name": "difficulty",
30              "type": "difficulty"
31            }
32          ]
33        },
34        {
35          "name": "gametype",
36          "type": "gametype",
37          "samples": [
38            {
39              "name": "gametype"
40            }
41          ]
42        }
43      ],
44      "samples": [
45        {
46          "utterance": "iniciar o jogo no modo (difficulty) e (gametype)",
47          "samples": [
48            "(gametype)",
49            "(difficulty)",
50            "(difficulty) (gametype)"
51          ]
52        }
53      ]
54    }
55  ]
56}

```

The footer includes 'Feedback', 'Errors and Warnings', and 'English (US)'.

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 se encontra a seguir: <https://aws.amazon.com>

Os passos são descritos a partir de X até a X.

Figura X – Página inicial da AWS Amazon

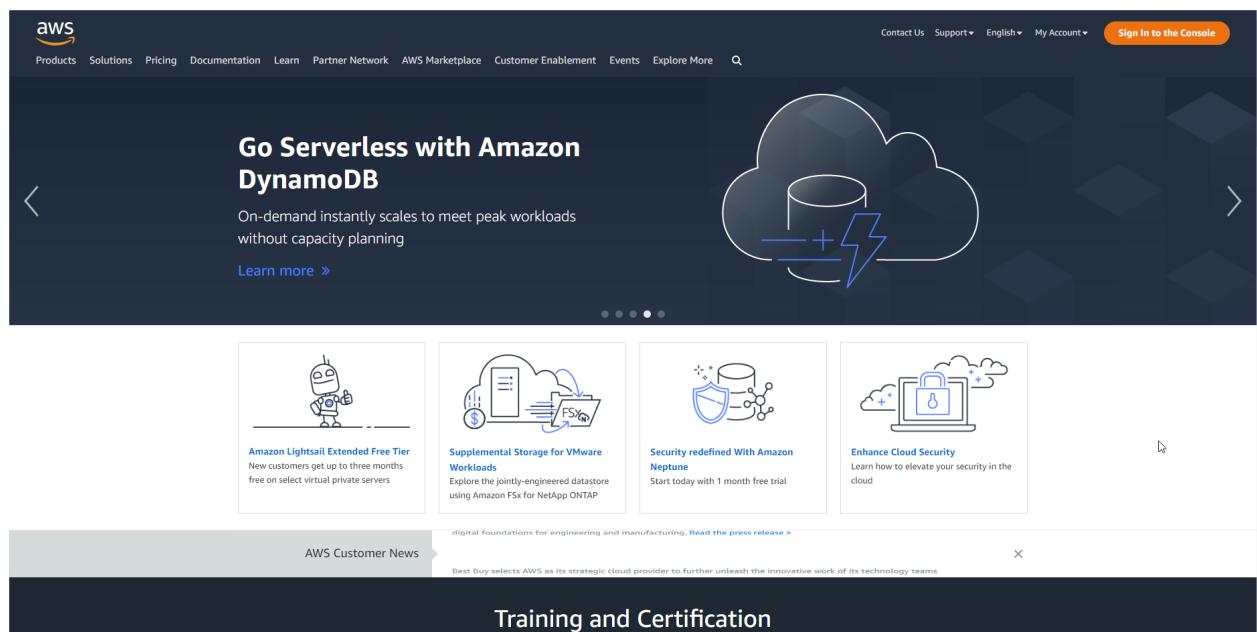


Figura X – Página de login da AWS Amazon

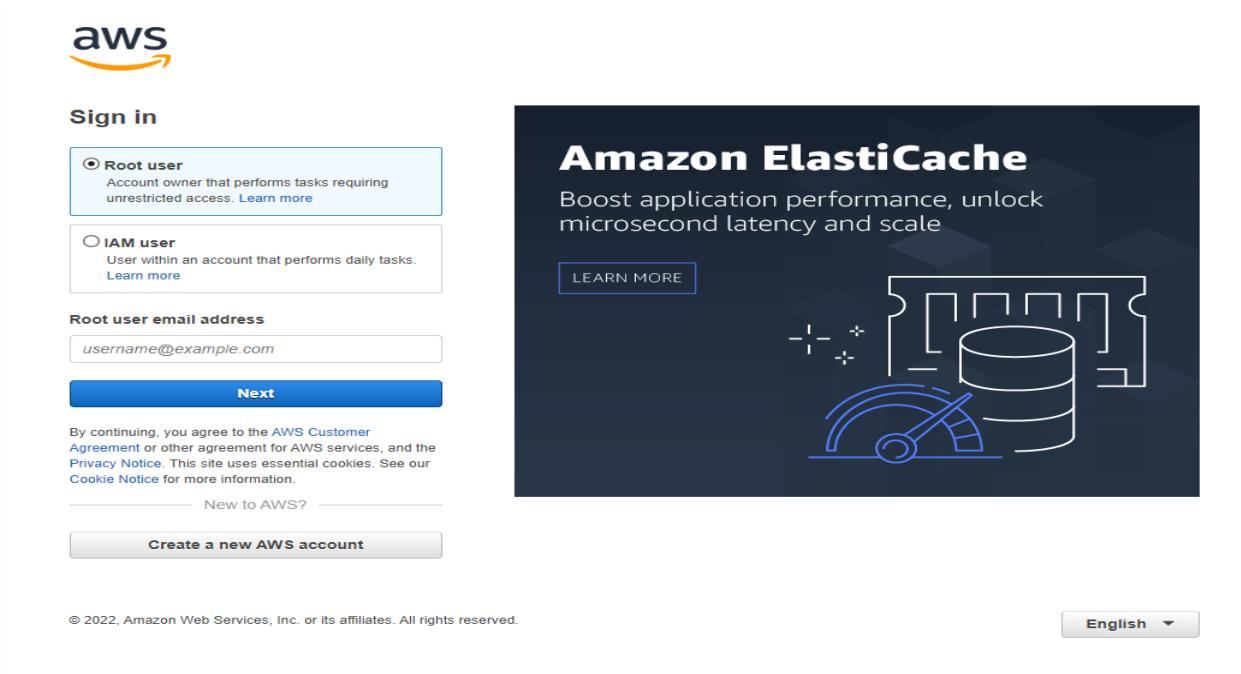


Figura X – Selecionar o serviço Lambda

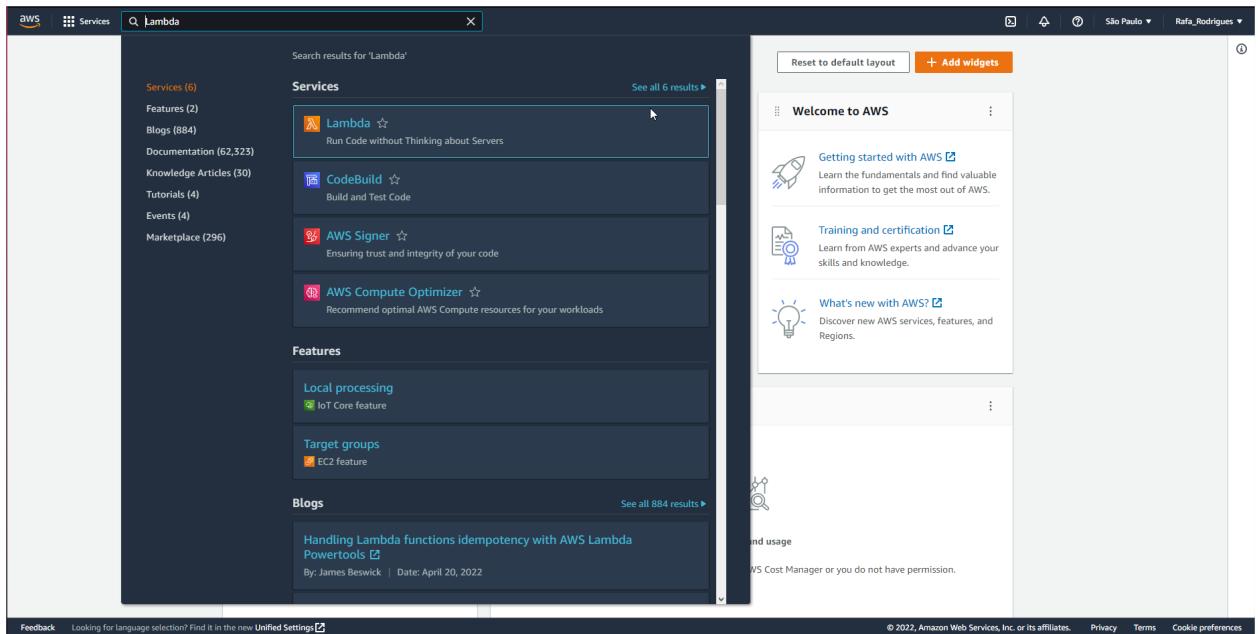


Figura X – Criar a função

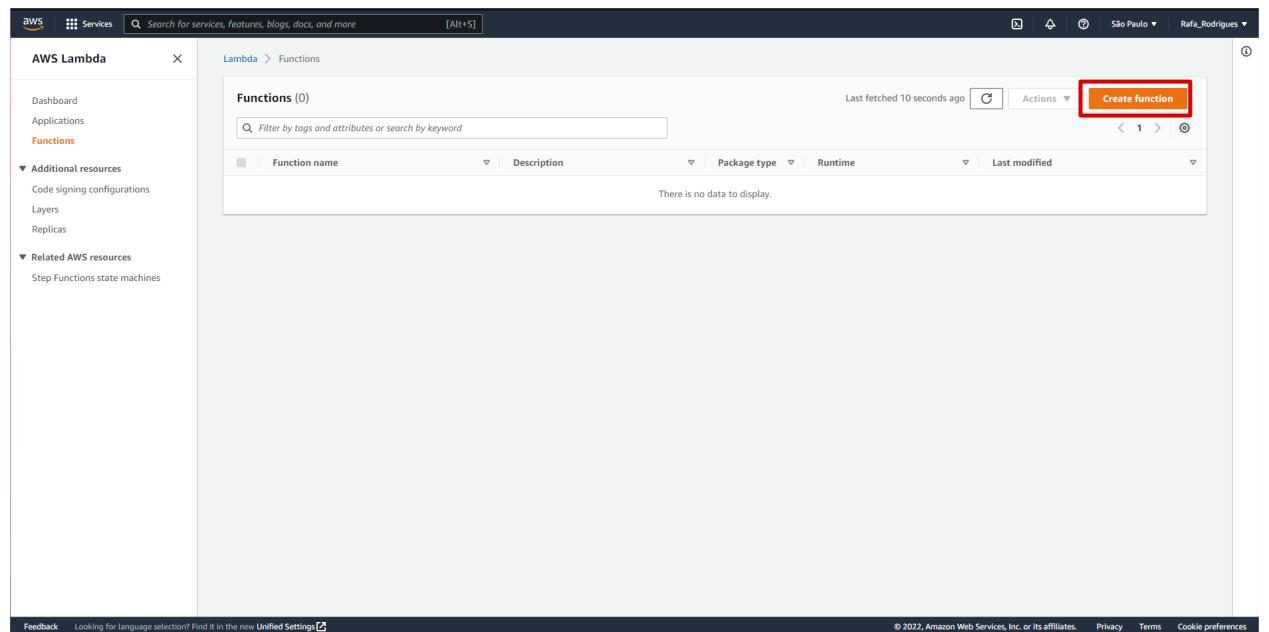


Figura X – Escolher um nome e linguagem de programação

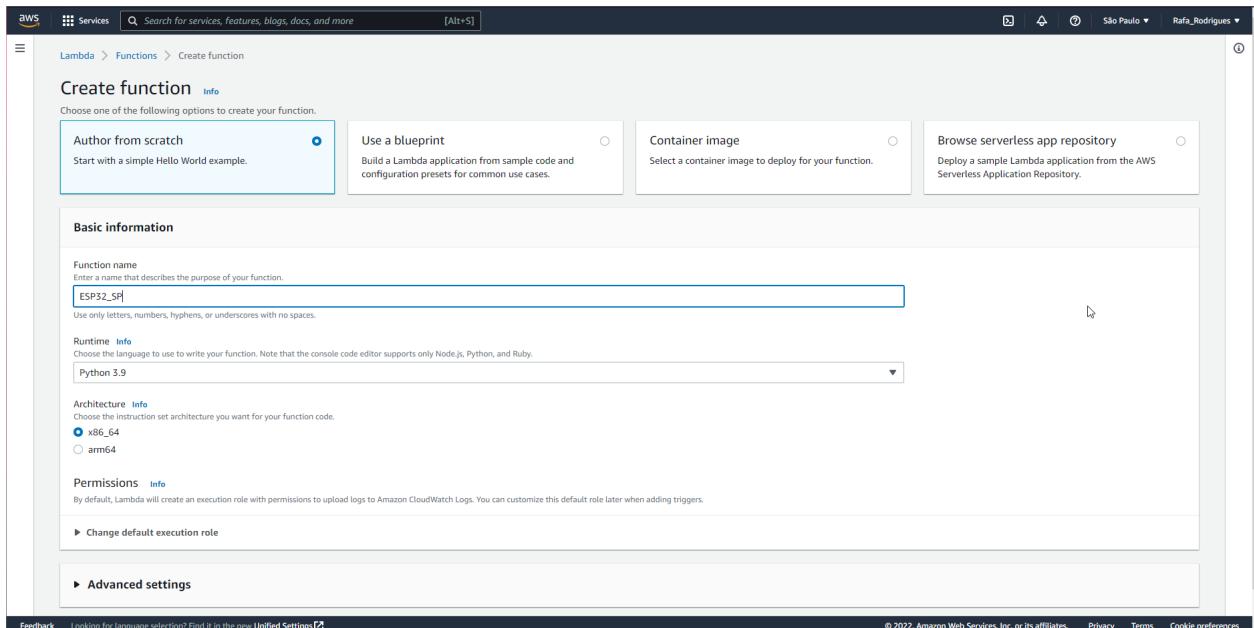
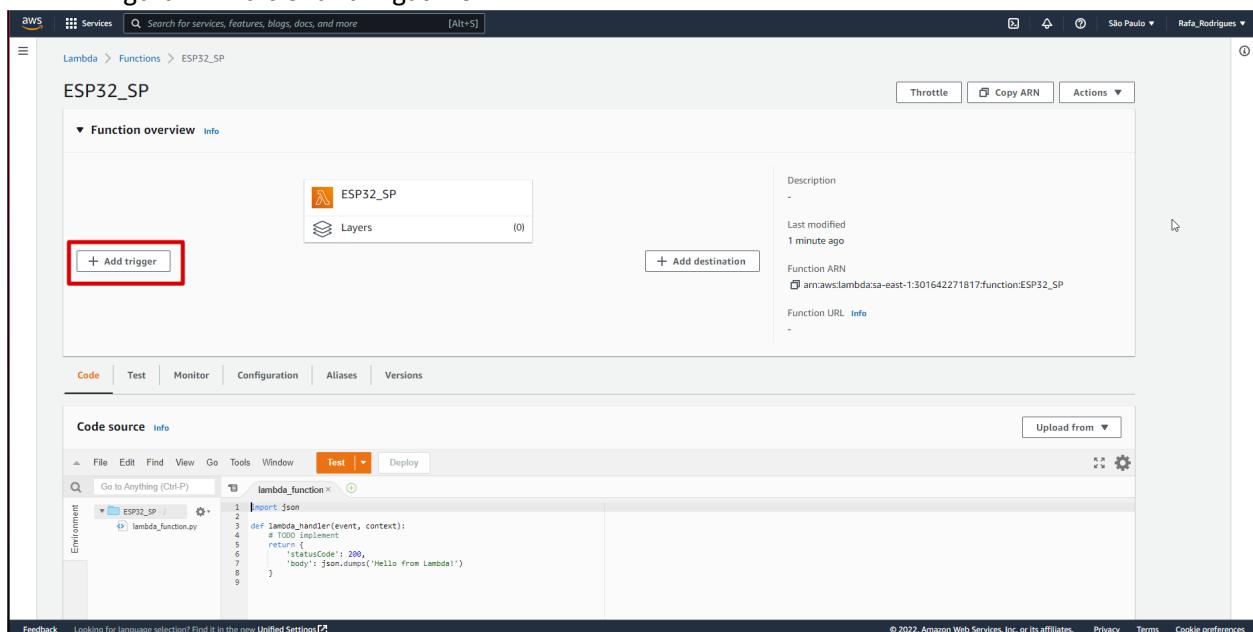


Figura X – Adicionar um gatilho



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

Figura X – Utilizar a opção “Copy to clipboard”

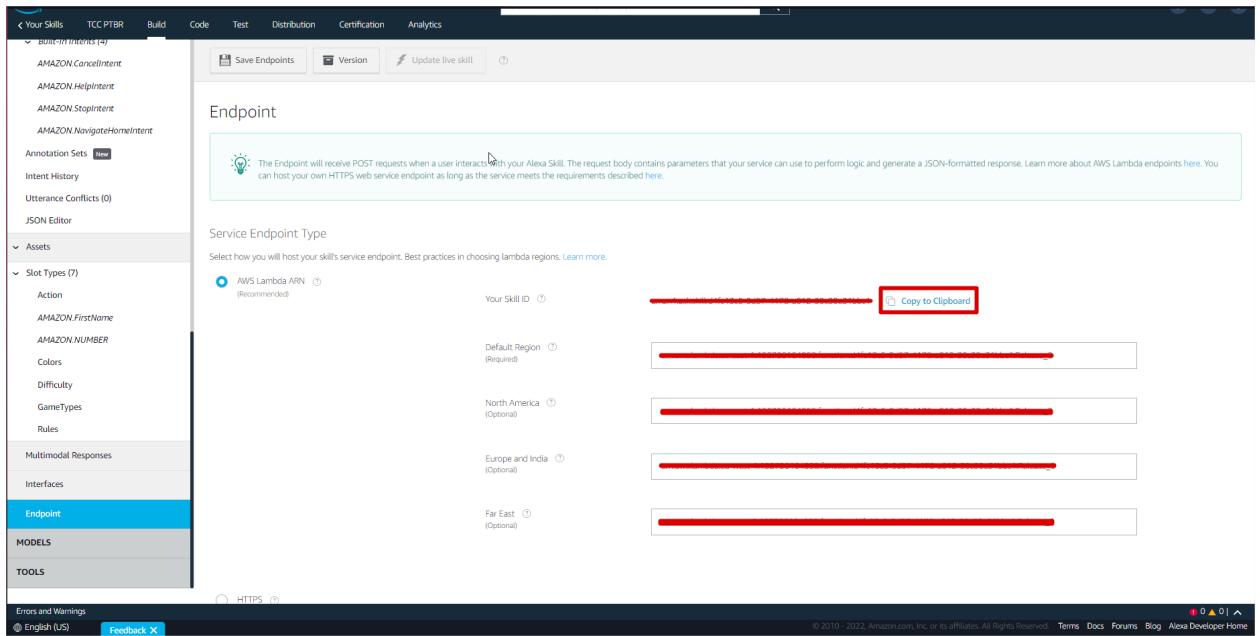
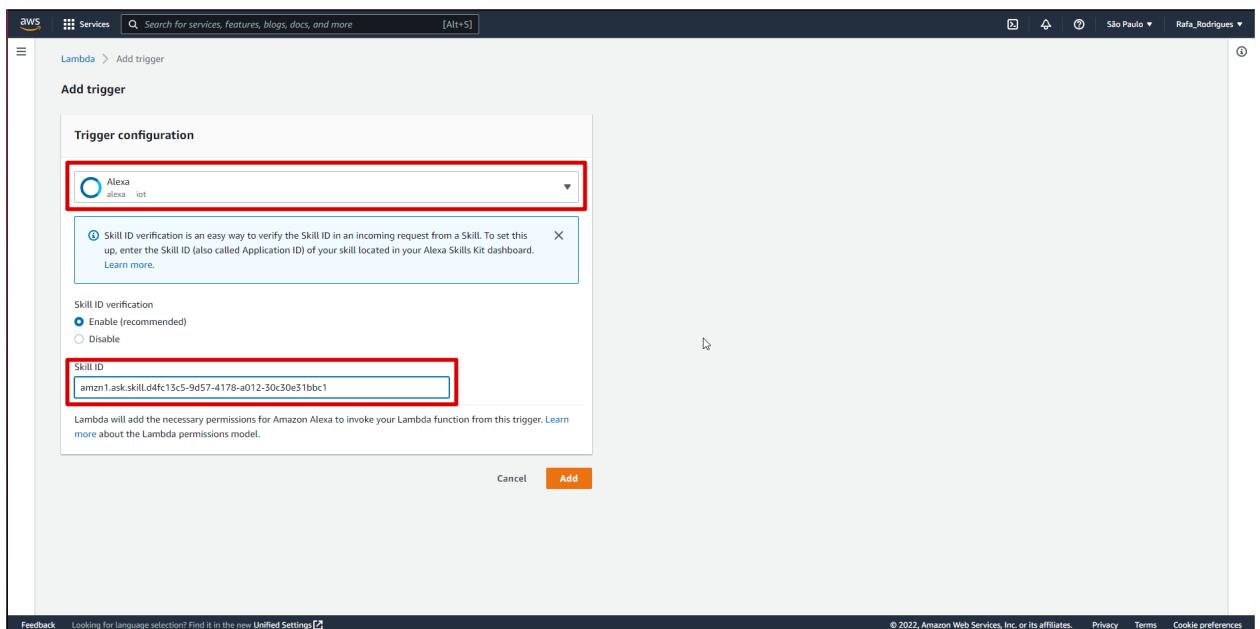
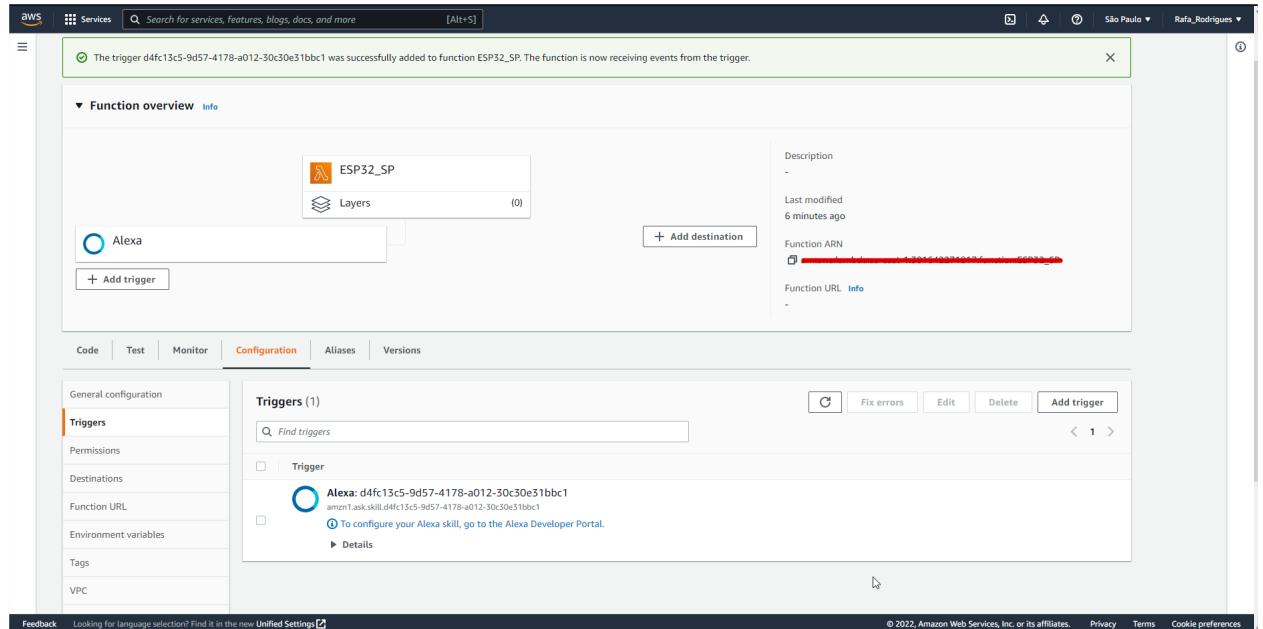


Figura X – Escolher “Alexa IoT” e colar o código copiado anteriormente no “Skill ID”

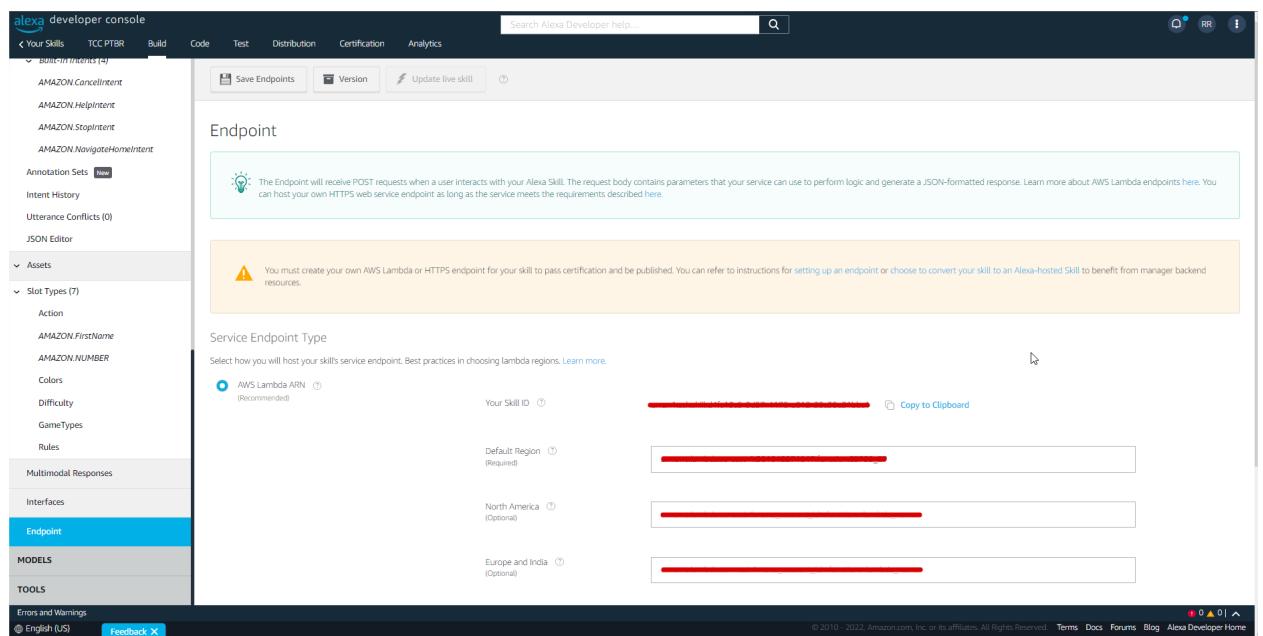


Será necessário realizar o processo inverso, copiar o código ARN da função Lambda e colar no endpoint da Skill para ela conseguir acessar o backend. A Figura X aponta onde se obtém na função Lambda. Na Figura X mostra onde deve ser inserido o ID na Skill. Após a inserção do ID, clicar em salvar.

**Figura X – Copiar o “Function ARN”**



**Figura X – Copiar 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 X e da Figura X.

**Figura X – Clicar no hiperlink**

Figura X – Expandir e editar a política

```

1 - [{
2 -   "Version": "2012-10-17",
3 -   "Statement": [
4 -     {
5 -       "Effect": "Allow",
6 -       "Action": "logs>CreateLogGroup",
7 -       "Resource": "arn:aws:logs:sa-east-1:301642271817:log-group:/aws/iot/*"
8 -     },
9 -     {
10 -       "Effect": "Allow",
11 -       "Action": [
12 -         "logs>CreateLogStream",
13 -         "logs:PutLogEvents"
14 -       ],
15 -       "Resource": [
16 -         "arn:aws:logs:sa-east-1:301642271817:log-stream:/aws/iot/*"
17 -       ]
18 -     }
19 -   ]
20 - }]
  
```

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 X, incluir o código de permissão do AWS IoT seguindo o exemplo do Quadro X.

Figura X – Colar e salvar

The screenshot shows the AWS Lambda Policy Editor interface. At the top, there's a navigation bar with the AWS logo, 'Services' dropdown, a search bar ('Search for services, features, blogs, docs, and more'), and a keyboard shortcut '[Alt+S]'. Below the search bar, the title 'Edit AWSLambdaBasicExecutionRole-dfc7278b-8407-4c1e-b56d-ab4a1cb49240' is displayed. A progress bar indicates steps 1 and 2. A note below the title says: 'A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. Learn more'.

The main area contains two tabs: 'Visual editor' (selected) and 'JSON'. To the right of the tabs is a link 'Import managed policy'. The JSON code is as follows:

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "logs:CreateLogGroup",
8          "logs:CreateLogStream",
9          "logs:PutLogEvents"
10         ],
11        "Resource": "arn:aws:logs:*:*"
12      },
13      {
14        "Effect": "Allow",
15        "Action": [
16          "iot:/*"
17        ],
18        "Resource": "arn:aws:iot:*:*"
19      }
20    ]
21  }

```

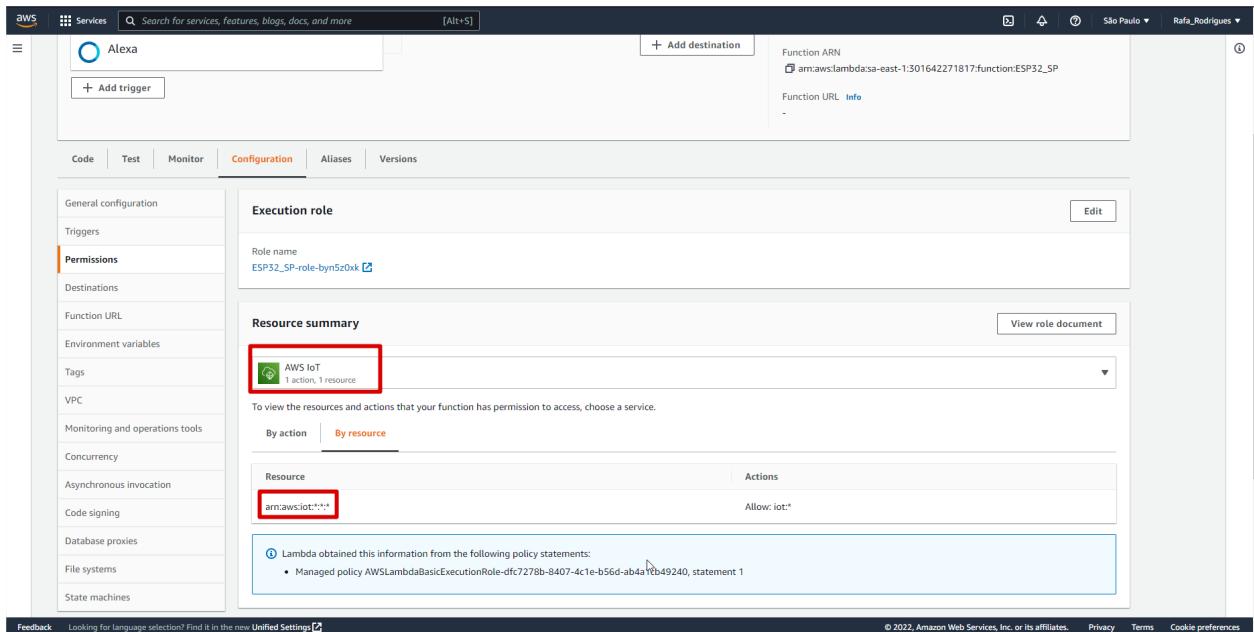
Below the JSON code, status indicators show: Security: 0, Errors: 0, Warnings: 0, Suggestions: 0. At the bottom right are 'Cancel' and 'Review policy' buttons. The footer includes links for 'Feedback', 'Language selection', 'Unified Settings', '© 2022, Amazon Web Services, Inc. or its affiliates.', 'Privacy', 'Terms', and 'Cookie preferences'.

Quadro X – Política utilizada

Código
<pre>{   "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:*:*"     }   ] }</pre>

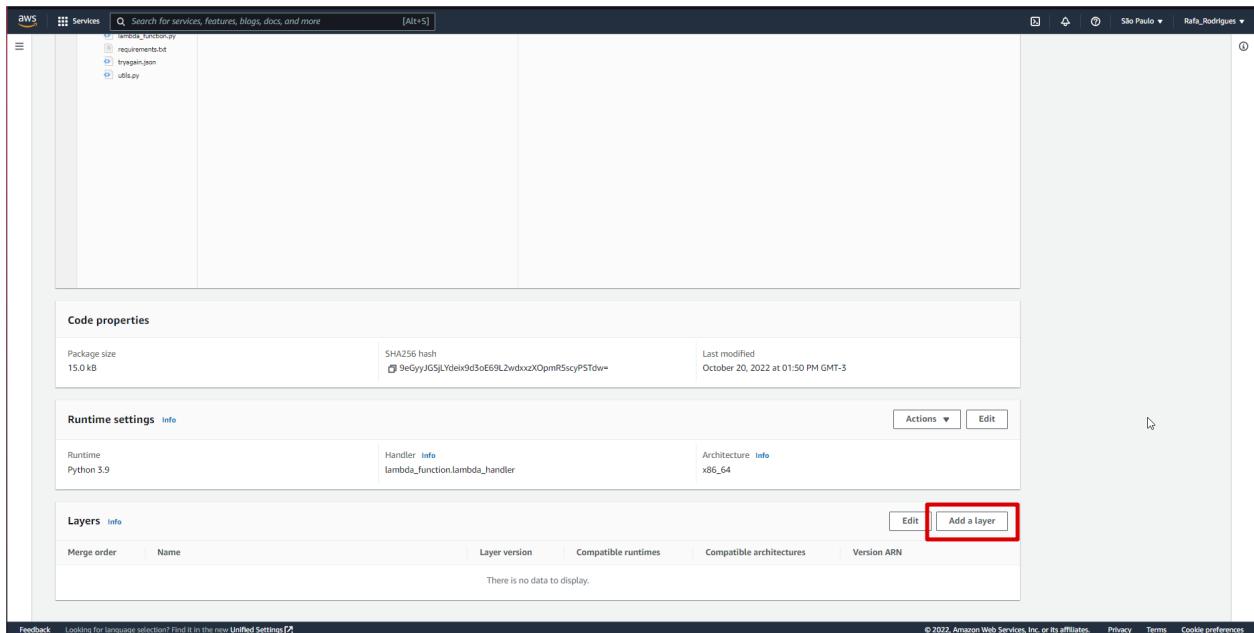
Após salvar e revisar a política a página deverá ficar como figura X.

Figura X – 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 X – 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 X – Especificando a camada

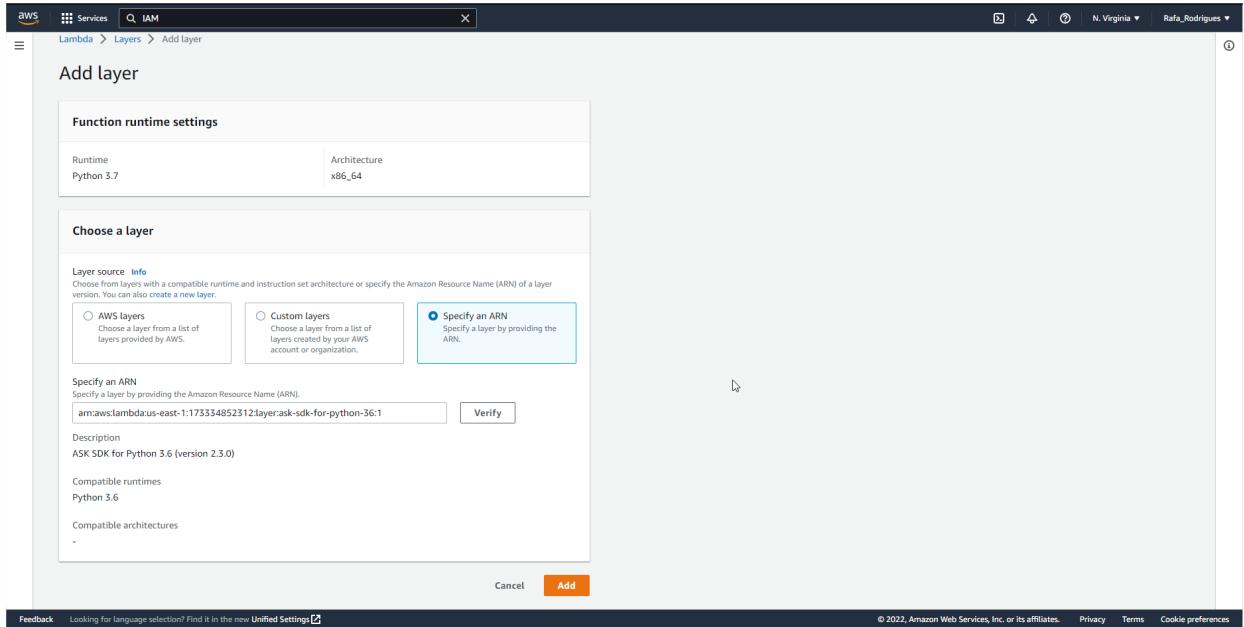
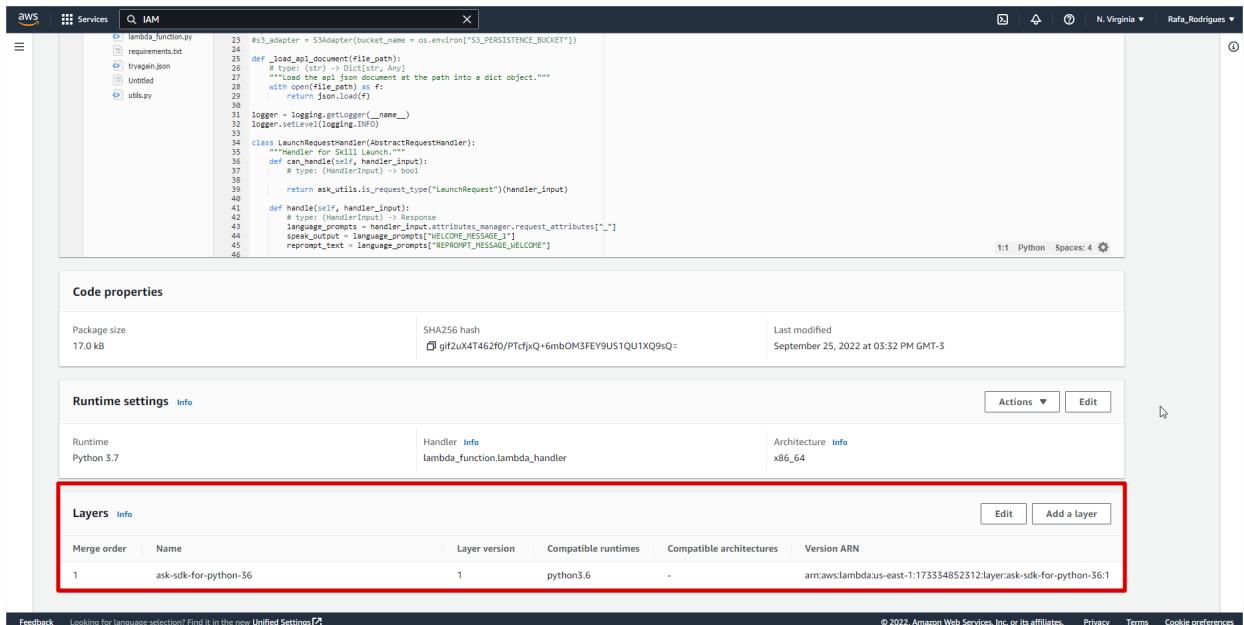


Figura X – Como ficará a camada



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 X – Código do backend

aws Services Search for services, features, blogs, docs, and more [Alt+S]

Successfully updated the function ESP32\_SP.

ESP32\_SP Layers (0)

Alexa + Add destination + Add trigger

Description - Last modified 27 minutes ago Function ARN arn:aws:lambda:sa-east-1:301642271817:function:ESP32\_SP Function URL Info -

Code Test Monitor Configuration Aliases Versions

Code source Info Upload from ▾

File Edit Find View Go Tools Window Test Deploy

Go to Anything (Ctrl+P) lambda\_function

Environment ESP32\_SP / SkillBuilder lambda\_function.py

```
1 # -*- coding: utf-8 -*-
2
3 # This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK for Python.
4 # Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,
5 # session persistence, api calls, and more.
6 # This sample is built using the handler classes approach in skill builder.
7 from ask_sdk_core.dispatch_components import AbstractRequestHandler, AbstractExceptionHandler, AbstractResponseInterceptor, AbstractRequestInterceptor
8 from ask_sdk_core.dispatch_components import (AbstractRequestHandler, AbstractExceptionHandler, AbstractResponseInterceptor)
9 from ask_sdk_core.handler_input import HandlerInput
10 from ask_sdk_core.utils import get_supported_interfaces
11 from ask_sdk_dynamodb.adapter import DynamoDbAdapter
12 from ask_sdk_core.handler_input import HandlerInput
13 from ask_sdk_core.utils import get_supported_interfaces
14 from ask_sdk_core.handler_input import HandlerInput
15 from ask_sdk_core.model import Response
16 from ask_sdk_core.model import Response
17 import logging
18 import os
19 import json
20 import json
21 import ask_sdk_core.utils as ask_utils
22 import ask_sdk_core.utils as ask_utils
23 #S3_adapter = S3Adapter(bucket_name = os.environ["S3_PERSISTENCE_BUCKET"])
```

Feedback Looking for language selection? Find it in the new Unified Settings ▾ © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

## 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 XXXX 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. (XXXX,2022).

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>

<https://developer.amazon.com/en-US/docs/alexa/alexa-design/apl.html#:~:text=Alexa%20has%20a%20visual%20design,ands%20engaging%20to%20the%20customer>. – Referência

## 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 X – Página inicial DynamoDB

The screenshot shows the AWS DynamoDB service dashboard. On the left, there's a sidebar with options like 'Dashboard', 'Tables' (which is selected), 'Explore items', 'PartiQL editor', 'Exports to S3', 'Imports from S3', 'Reserved capacity', and 'Settings'. Below that is a section for 'DAX' with 'Clusters', 'Subnet groups', 'Parameter groups', and 'Events'. The main area is titled 'Tables (1)' and shows a table with one item. The table has columns: Name, Status, Partition key, Sort key, Indexes, Read capacity mode, Write capacity mode, Size, and Table class. The item in the table is 'users\_score' with status 'Active', partition key 'username (\$)', sort key ' - ', 0 indexes, provisioned with auto scaling (1) for both read and write, 107 bytes size, and 'DynamoDB Standard' as the table class. There are buttons for 'Actions', 'Delete', and 'Create table' at the top right of the table list.

Criar a tabela informando o nome desejado.

Figura X – Criação de tabela

This screenshot shows the 'Create table' wizard in the AWS DynamoDB console. The first step, 'Table details', is active. It asks for a 'Table name' which is currently 'Enter name for table'. It also defines a 'Partition key' as 'String' type and a 'Sort key' as 'String' type. In the 'Table settings' section, there are two radio buttons: 'Default settings' (selected) and 'Customize settings'. The 'Default settings' option is described as the fastest way to create a table and allows modification later. The 'Customize settings' option is described as using advanced features to make DynamoDB work better. At the bottom, there are 'Next Step' and 'Cancel' buttons.

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

Figura X - Permissão

The screenshot shows the AWS Lambda function configuration page. The left sidebar lists various service options like Code, Test, Monitor, Configuration, Aliases, and Versions. The 'Configuration' tab is selected. In the main area, under 'Execution role', the 'Role name' field is highlighted with a red box, containing the value 'ESP32-role-vm0fqmf3'. Below it, the 'Resource summary' section shows a single resource: 'AWS Application Auto Scaling' with 7 actions and 1 resource. A table below details the permissions granted to this role.

Resource	Actions
All resources	Allow: application-autoscaling:DeleteScalingPolicy Allow: application-autoscaling:DeregisterScalableTarget Allow: application-autoscaling:DescribeScalableTargets Allow: application-autoscaling:DescribeScalingActivities Allow: application-autoscaling:DescribeScalingPolicies Allow: application-autoscaling:PutScalingPolicy Allow: application-autoscaling:RegisterScalableTarget

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

Figura X – Página de permissões

The screenshot shows the AWS IAM Roles page. The left sidebar includes 'Identity and Access Management (IAM)', 'Access management', 'Access reports', and 'Related consoles'. The 'Roles' section is selected. The main area shows a role named 'ESP32-role-vm0fqmf3'. The 'Permissions' tab is active. A modal window titled 'Introducing the new IAM roles experience' is displayed, stating 'We've redesigned the IAM roles experience to make it easier to use. Let us know what you think.' At the bottom right of this modal, the 'Attach policies' button is highlighted with a red box. The 'Permissions policies' table lists two managed policies: 'AWSLambdaBasicExecutionRole-226ee89d-4cc5-4ca2-8df0-5736a11fb0e8' (Customer managed) and 'AmazonDynamoDBFullAccess' (AWS managed).

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

Figura X – Adicionando nova permissão

The screenshot shows the AWS IAM service interface. In the top navigation bar, 'Services' is selected. Below it, the path 'IAM > Roles > ESP32 > Add permissions' is shown. A banner at the top says 'Introducing the new IAM roles experience' with a link to learn more. The main area is titled 'Attach policy to ESP32' and shows 'Current permissions policies (1)'. Below this, a section titled 'Other permissions policies (773)' contains a search bar with the query 'dynamodb' and a result count of '4 matches'. A red box highlights the first result, 'AmazonDynamoDBFullAccess', which is listed as an 'AWS managed' policy. The table columns are 'Policy name', 'Type', and 'Description'. The 'Description' column for the highlighted policy states: 'Provides full access to Amazon DynamoDB via the AWS Management Console.' At the bottom right are 'Cancel' and 'Attach policies' buttons.

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

Figura X – Lista de permissões

The screenshot shows the AWS Lambda service interface. On the left, a sidebar lists various configuration options: Triggers, Permissions (which is currently selected), Destinations, Function URL, Environment variables, Tags, VPC, Monitoring and operations tools, Concurrency, Asynchronous invocation, Code signing, Database proxies, File systems, and State machines. The main panel shows the 'Execution role' for the function 'ESP32-role-vm0fqmf3'. Under 'Resource summary', there is a list of services and their actions/resources. A red box highlights the entry for 'Amazon DynamoDB', which is listed under the 'AWS Application Auto Scaling' service. Other entries include AWS IoT, AWS Key Management Service, AWS Lambda, AWS Resource Groups, Amazon CloudWatch, Amazon CloudWatch Logs, Amazon DynamoDB Accelerator (DAX), Amazon EC2, Amazon Kinesis, Amazon Resource Group Tagging API, and Amazon SNS. At the bottom of the list is a 'Find policy statements' button.

## 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.

```
--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.

```
--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
        ReturnValue="UPDATED_NEW"
    )
    return response['Attributes']
```

### *Consultar*

Exemplo para retornar um dado na tabela.

```
--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 "bhagiii" == users_l['username']:
            users_test += users_l['username'] + " -> " + users_l['score']
    print(users_test)
    return None
```

### *Excluir*

Exemplo para excluir um dado na tabela.

```
--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']
```

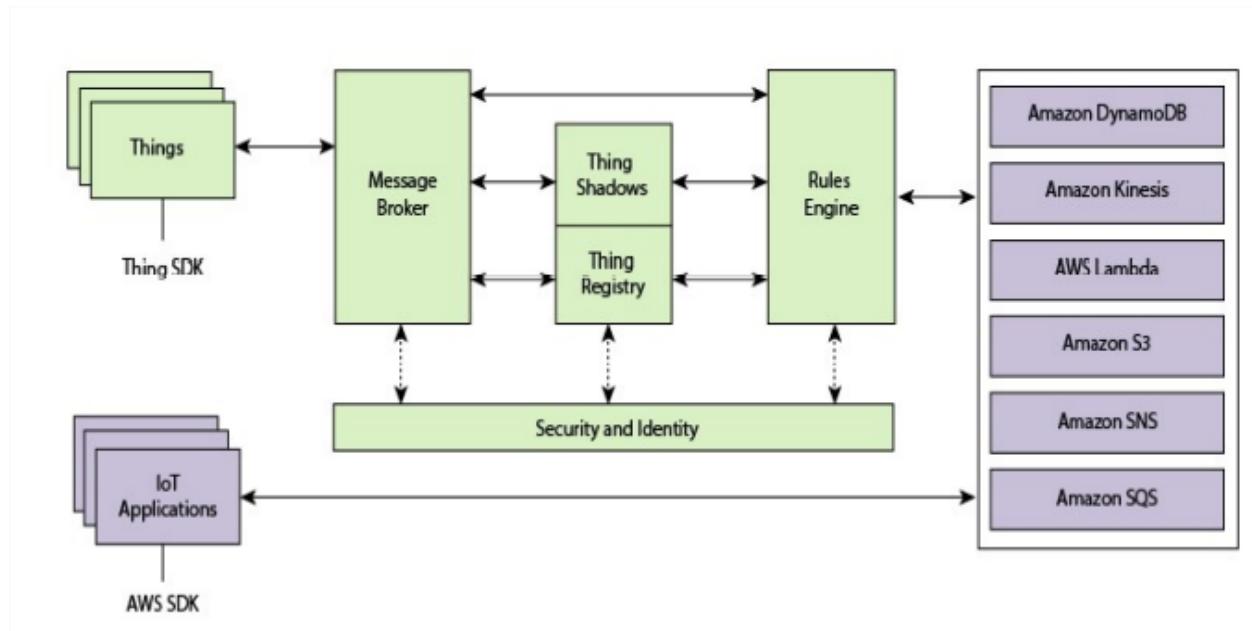
### *Criação de uma “coisa”*

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 X se encontra uma exemplificação desta arquitetura.

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

Figura X – Estrutura do AWS IoT

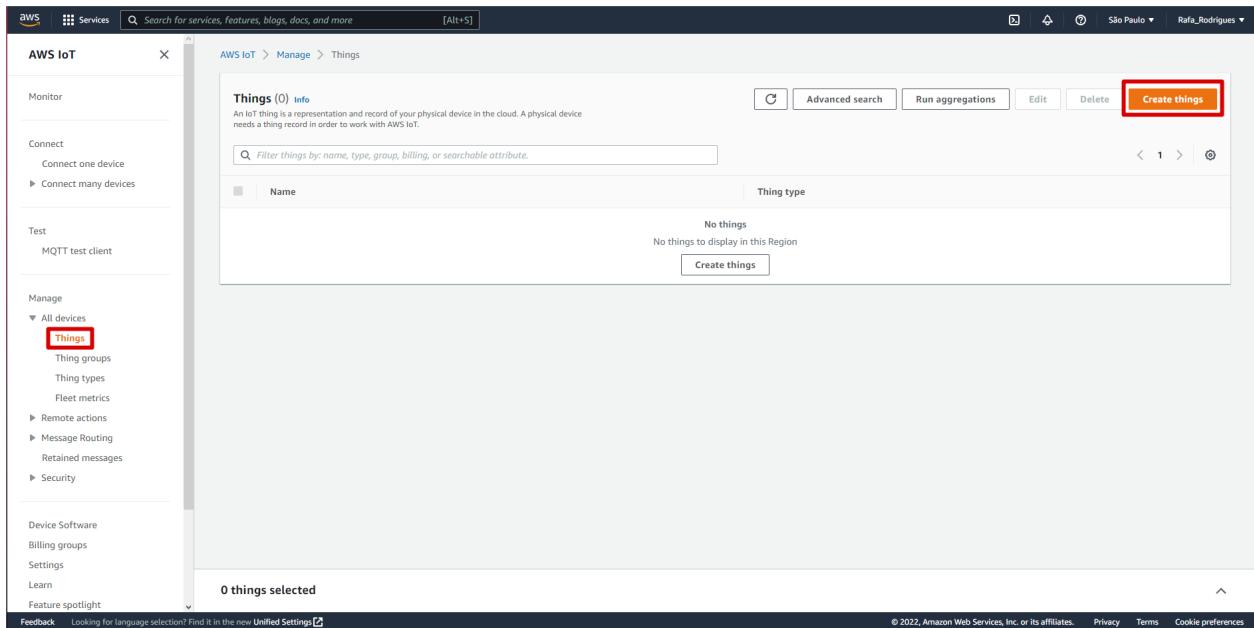


A partir da Figura X está descrito o passo-a-passo de como criar uma “coisa”, ou um novo dispositivo, à função em AWS IoT, terminando na Figura X.

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.

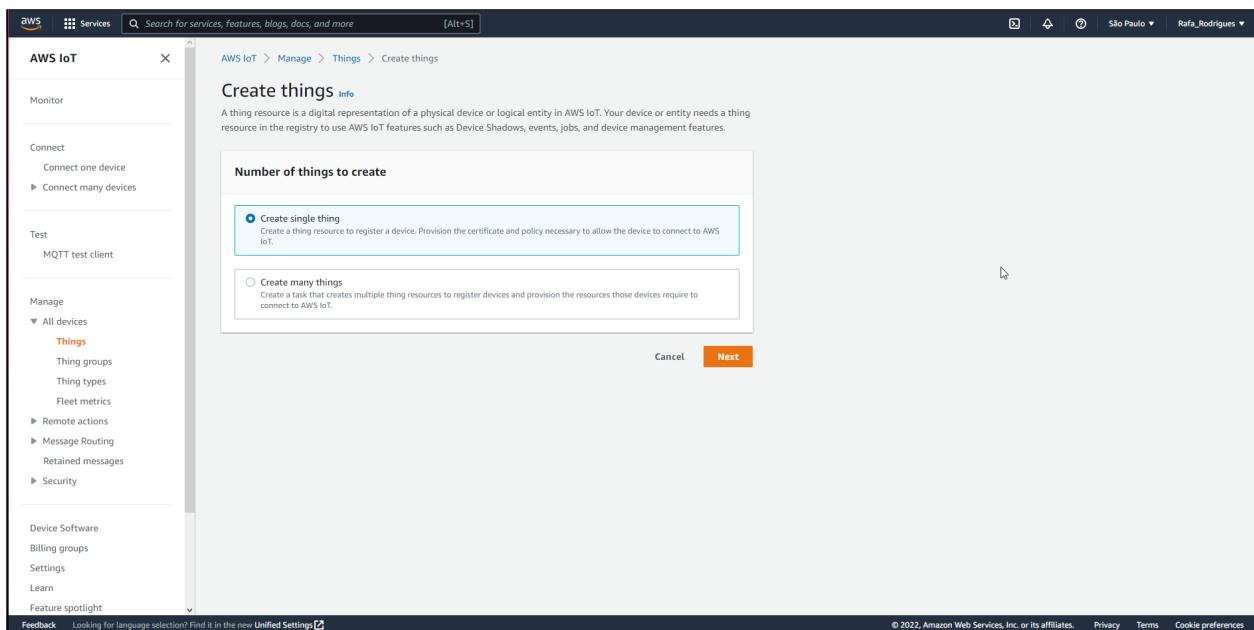
Passo 1: Acessar a página do IoT Core, acessar o item “Things” e utilizar a opção “Create things”.

Figura X – Página inicial AWS IoT



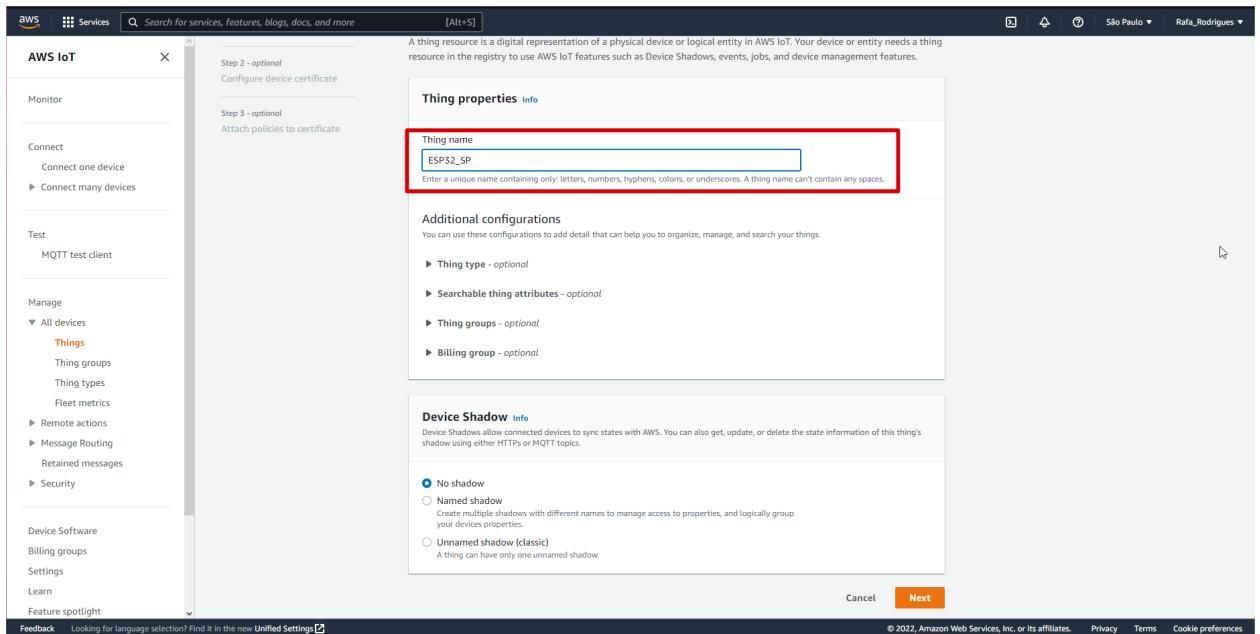
Passo 2: Utilizar a opção “Create single thing” e clicar em “Next”.

Figura X – Criação da “coisa”



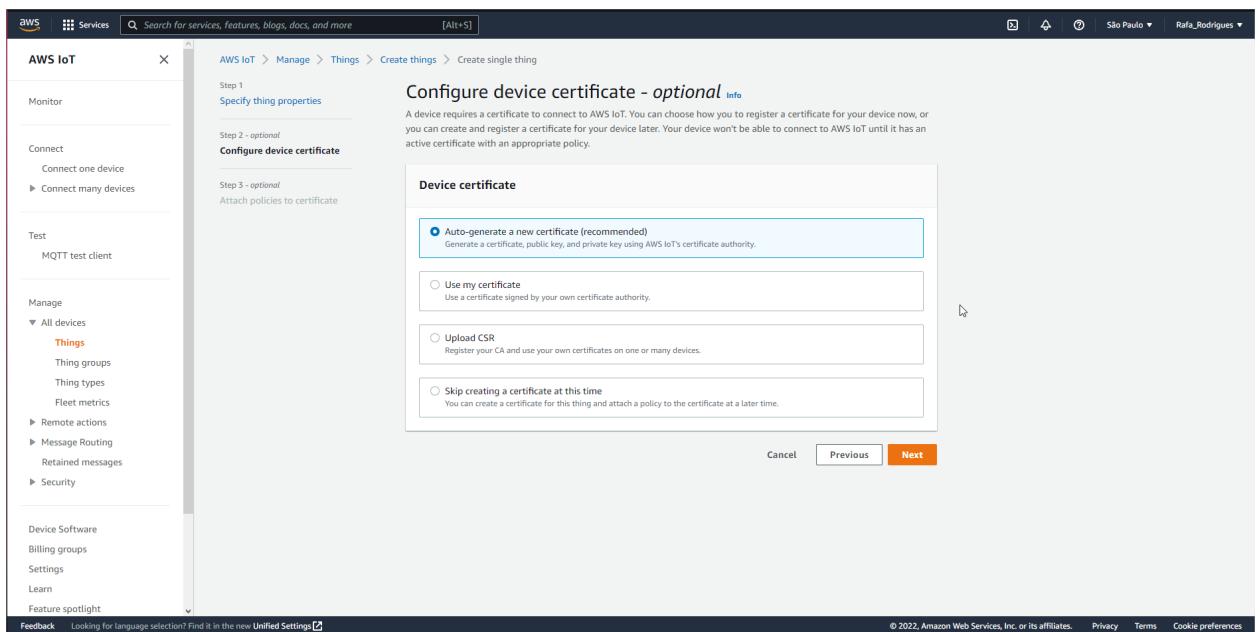
Passo 3: Inserir um novo nome e demais valores são opcionais.

Figura X – Propriedades da “coisa”



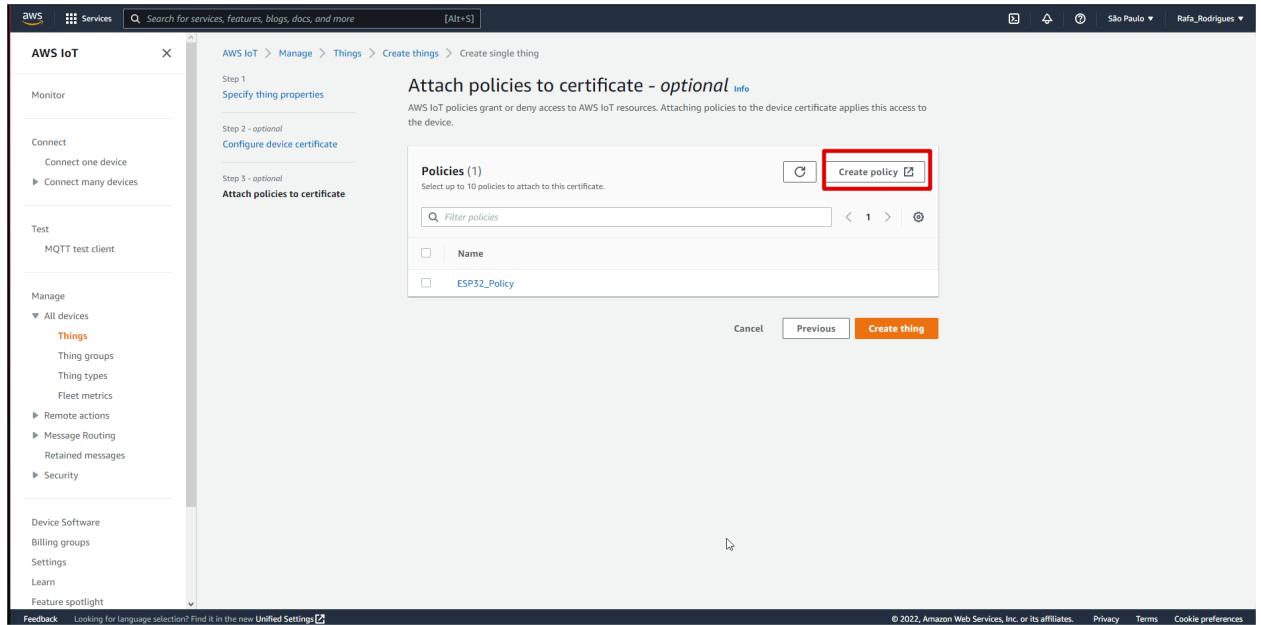
Passo 4: Utilizar a opção “Auto-generate a new certificate (recommended)” e clicar em “Next”.

Figura X – Escolha do certificado



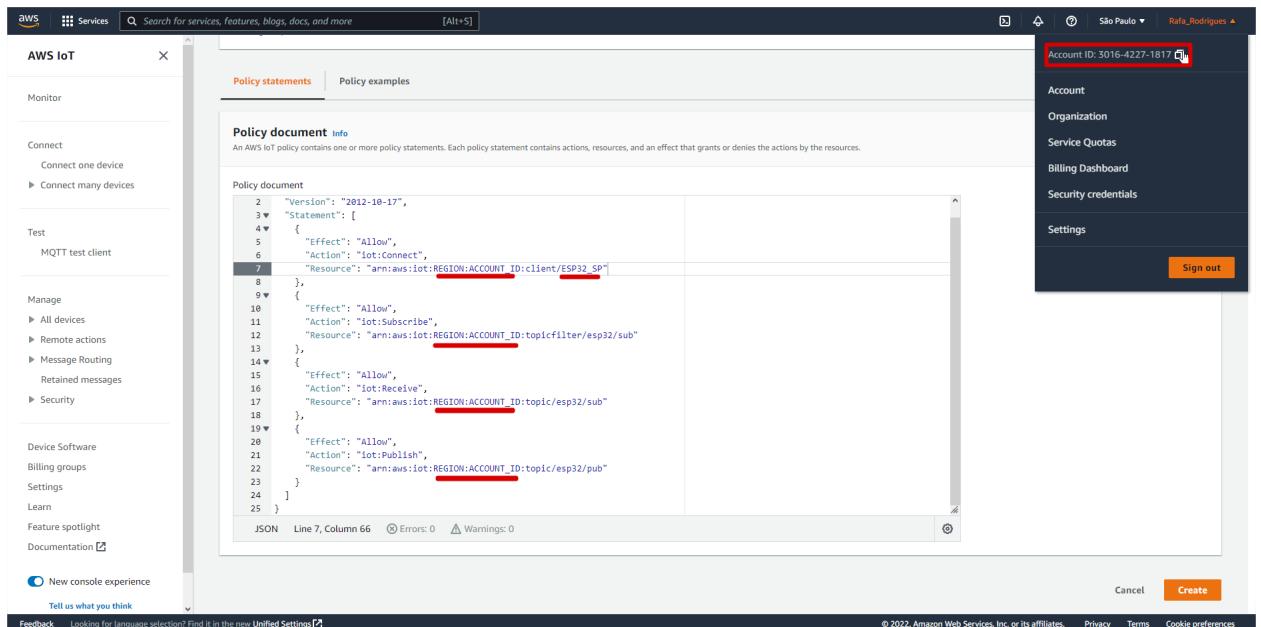
Passo 5: Criar uma política.

Figura X – Política da “coisa”



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

Figura X – Criação de política



Quadro X – 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/pub" }, { "Effect": "Allow", "Action": "iot:Publish", "Resource": "arn:aws:iot:REGION:ACCOUNT_ID:topic/esp32/pub" } ] }

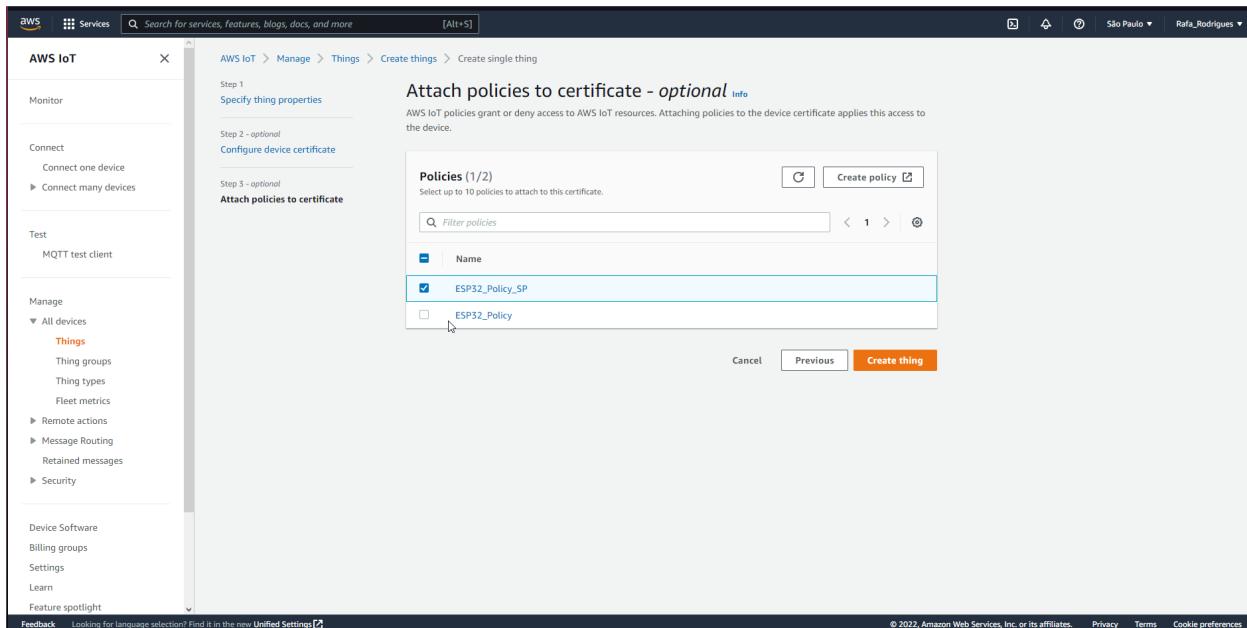
```

    "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”

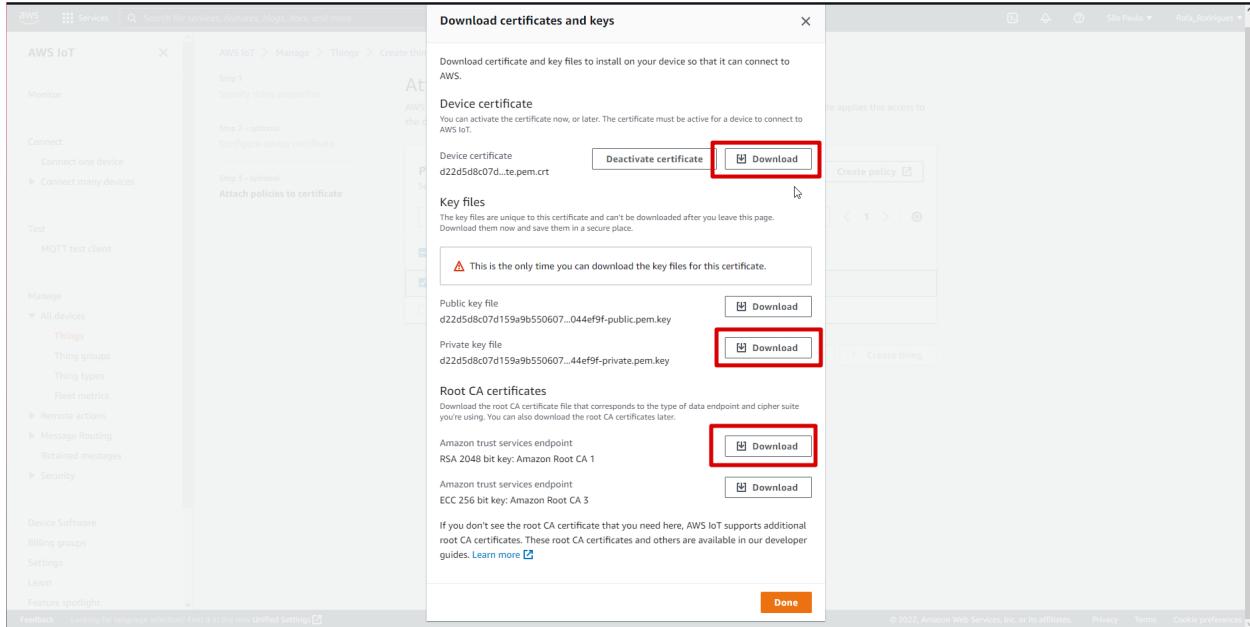
Figura X – Vinculo da política



#### 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 grifados em vermelho na figura X.

Figura X – Download dos certificados



## Testes MQTT

### 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 X.
- 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 X – Endpoint IoT Core.

The screenshot shows the AWS IoT Core Settings page. On the left, there's a navigation sidebar with various options like All devices, Things, Thing groups, Thing types, Fleet metrics, Remote actions, Message Routing, Retained messages, Security, and more. The 'Settings' option is highlighted. The main content area has two main sections: 'Device data endpoint' and 'Domain configurations'. The 'Device data endpoint' section contains a note about using the device data endpoint for REST API access and an 'Endpoint' field. The 'Domain configurations' section includes a 'Create domain configuration' button and a note about migrating devices to AWS IoT Core. At the bottom, there's a 'Logs' section with a 'Manage logs' button.

## Ferramenta de testes MQTT Lambda

Após acessar a página do IoT Core, acessar o item “MQTT test client”. E configurar os tópicos conforme figura X e X.1.

Figura X – Configuração do tópico MQTT

The screenshot shows the AWS IoT Core MQTT test client configuration page. The left sidebar has a 'Test' section with 'MQTT test client' selected. The main area is titled 'MQTT test client' and describes how it can be used to monitor MQTT messages. It features two tabs: 'Subscribe to a topic' (selected) and 'Publish to a topic'. Under 'Subscribe to a topic', there's a 'Topic filter' input field containing 'esp32/pub' with a red box around it, and a 'Subscribe' button with a red box around it. Below this, the 'Subscriptions' section shows a single entry 'esp32/pub' with a red box around it. A note says 'No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here.'

Figura X.1 – Configuração do tópico MQTT

The screenshot shows the AWS IoT MQTT test client interface. On the left, a sidebar menu includes options like Monitor, Connect, Test (selected), and MQTT test client. Under Test, there are sections for Manage (All devices, Things, Thing groups, Thing types, Fleet metrics), Remote actions, Message Routing, Retained messages, Security (Intro, Certificates, Policies, Certificate authorities, Role Aliases, Authorizers), and a Feedback link. The main content area is titled "MQTT test client" and shows a "Subscribe to a topic" section with a red box around the topic name input field containing "Q\_esp32/sub". Below it is a "Message payload" field with the JSON message: {"message": "Hello from AWS IoT console"}. There is also an "Additional configuration" section with a "Publish" button. To the right, a "Subscriptions" section shows a single entry for "esp32/pub" with a red box around it. A message box below says "No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here." At the bottom, there are buttons for Pause, Clear, Export, and Edit.

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 X – Envio da mensagem ao Arduino

This screenshot is identical to the one above, showing the AWS IoT MQTT test client interface. The "Topic name" field contains "Q\_esp32/sub", and the "Message payload" field contains the JSON message: {"sequence": "14,14,26,26,27,27,27,"}, {"delay": "500"}. The "Publish" button is highlighted with a red box. The "Subscriptions" section shows the "esp32/pub" entry, and the message box below it remains the same: "No messages have been sent to this subscription yet. Please send a message to this subscription to see messages here."

Figura X – Log de recebimento no Arduino

The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a window title AWS\_BULB\_SELECT\_SP | Arduino 1.8.20 hourly Build 2022/04/25 09:33. The main area contains the C++ code for the AWS BULB SELECT SP sketch. Below the code is a serial monitor window titled "COM3". The serial monitor displays the log of incoming data from the ESP32, which consists of a sequence of numbers separated by commas: "14, 14, 26, 26, 27, 27, 27,". The window also includes controls for Autoscroll, Show timestamp, and baud rate selection (No line ending, 9600 baud, Clear output).

```
File Edit Sketch Tools Help

AWS_BULB_SELECT_SP | Secretshh
ptr = strtok(strg_array, ","); // delimiter
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(1000);
    digitalWrite(strings[n], LOW);
    delay(1000);
}
publishReturnMessage();

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

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

Done uploading

Writing at 0x000094d1... (64 %)
Writing at 0x000094f1... (67 %)
Writing at 0x00009511... (71 %)
Writing at 0x00009531... (73 %)
Writing at 0x00009551... (76 %)
Writing at 0x00009571... (76 %)
Writing at 0x00009591... (82 %)
Writing at 0x000095b1... (82 %)
Writing at 0x000095d1... (88 %)
Writing at 0x000095e1... (88 %)
Writing at 0x000095f1... (93 %)
Writing at 0x00009691... (94 %)
Writing at 0x000096c1... (94 %)
Writing at 0x00009721... (100 %)
Wrote 847054 bytes (548923 compressed) at 0x00010000 in 9.1 seconds (effective 735.8 kbit/s)...
```

## Figura X – Log de sucesso IoT Core

The screenshot shows the AWS IoT Device Management console. On the left, a sidebar lists navigation options: Services, Monitor, Connect, Test, MQTT test client, Manage, Device Software, Billing groups, Settings, Learn, Feature spotlight, and Documentation. The main area is titled "AWS IoT" and shows a "Subscribe to a topic" interface. A search bar at the top has the placeholder "Search for services, features, blogs, docs, and more". Below it, a "Topic name" field contains "esp32/pub" with a placeholder "The topic name identifies the message. The message payload will be published to this topic with a Quality of Service (QoS) of 0.". An "Additional configuration" section includes a "Message payload" field with the following JSON content:

```
{  
  "sequence": "14, 14, 26, 26, 27, 27, 27,",  
  "delay": "500"  
}
```

Below this is a "Publish" button. To the right, under "Subscriptions", there is a table with one row for "esp32/pub". The table columns are "Topic" and "Actions". The "Topic" column shows "esp32/pub" and the "Actions" column shows "Pause", "Clear", "Export", and "Edit". The table also includes a "Last updated" timestamp: "October 20, 2022, 12:39:31 (UTC-03:00)". At the bottom of the page, there is a feedback section with the text "Tell us what you think" and a "Feedback" button.