

1. PRIMEIROS PASSOS

O primeiro passo para começar o desenvolvimento de nossa *skill* é acessar o seguinte endereço online: <https://developer.amazon.com/pt-BR/alexa/alexa-skills-kit>.

Após isso, devemos efetuar nosso cadastro no site.

The screenshot shows the Alexa Skills Kit developer page. At the top, there is a navigation bar with the Alexa logo, a search bar, and a 'Entrar' button. Below the navigation bar, there is a main banner with the text 'Novidades Prêmio Alexa de Acessibilidade!' and a 'Conheça os Vencedores' button. The main content area is titled 'Alexa Skills Kit' and contains the following text:

O que são Alexa Skills?

Skills são como aplicativos para Alexa e fornecem um novo canal para seu conteúdo ou serviço. As skills permitem que usuários usem suas vozes para realizar tarefas diárias, como verificar notícias, ouvir música, jogar um jogo e muito mais. Empresas e indivíduos podem publicar skills na Alexa Skills Store para alcançar e encantar clientes através de milhões de dispositivos habilitados para Alexa.

O Alexa Skills Kit (ASK) fornece APIs e ferramentas que você pode usar para criar skills de Alexa. Você também pode contratar uma agência especialista em Alexa para ajudar sua empresa a desenvolver uma skill.

Below the text, there is an image of various Alexa devices. At the bottom of the page, there is a footer with the text 'Português (Brasil)' and '© 2010-2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Termos Documentação Fóruns Blog Alexa Developer Home'.

The screenshot shows the Amazon Sign in page. At the top, there is the Alexa logo. Below it, there is a 'Sign in' section with the following fields:

Sign in

Email or mobile phone number

Password [Forgot your password?](#)

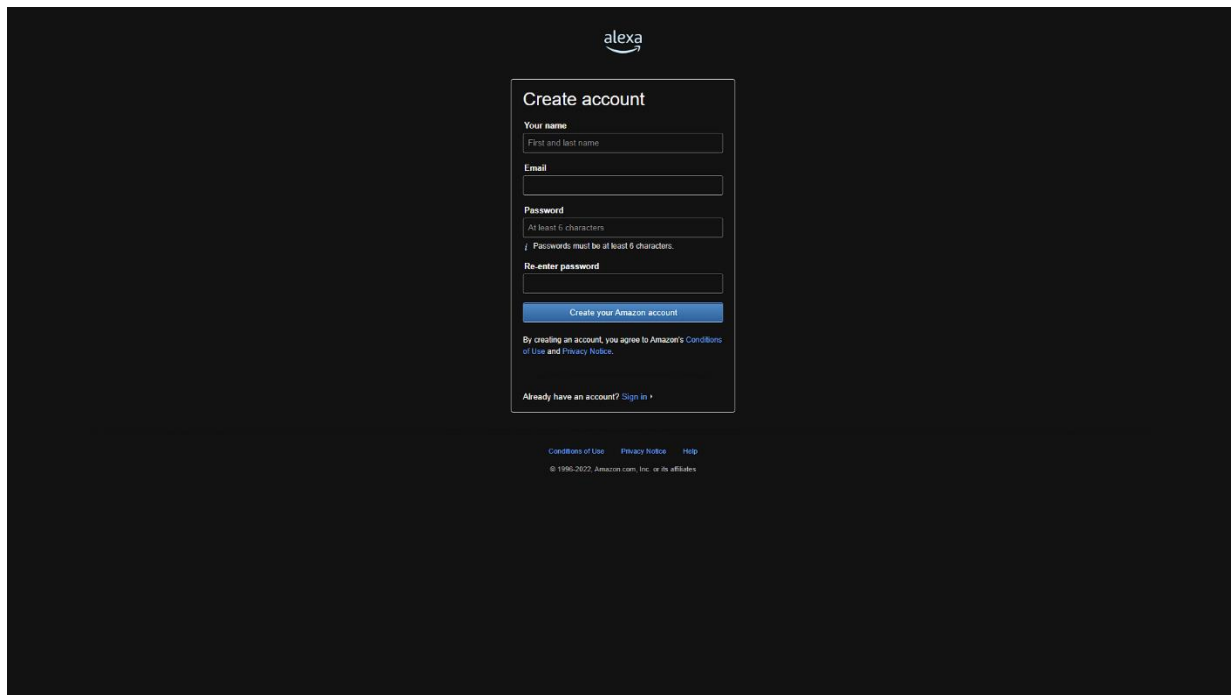
By continuing, you agree to Amazon's [Conditions of Use](#) and [Privacy Notice](#).

Keep me signed in. [Details](#)

[New to Amazon?](#)

At the bottom of the page, there is a footer with the text 'Conditions of Use Privacy Notice Help' and '© 1996-2022, Amazon.com, Inc. or its affiliates'.

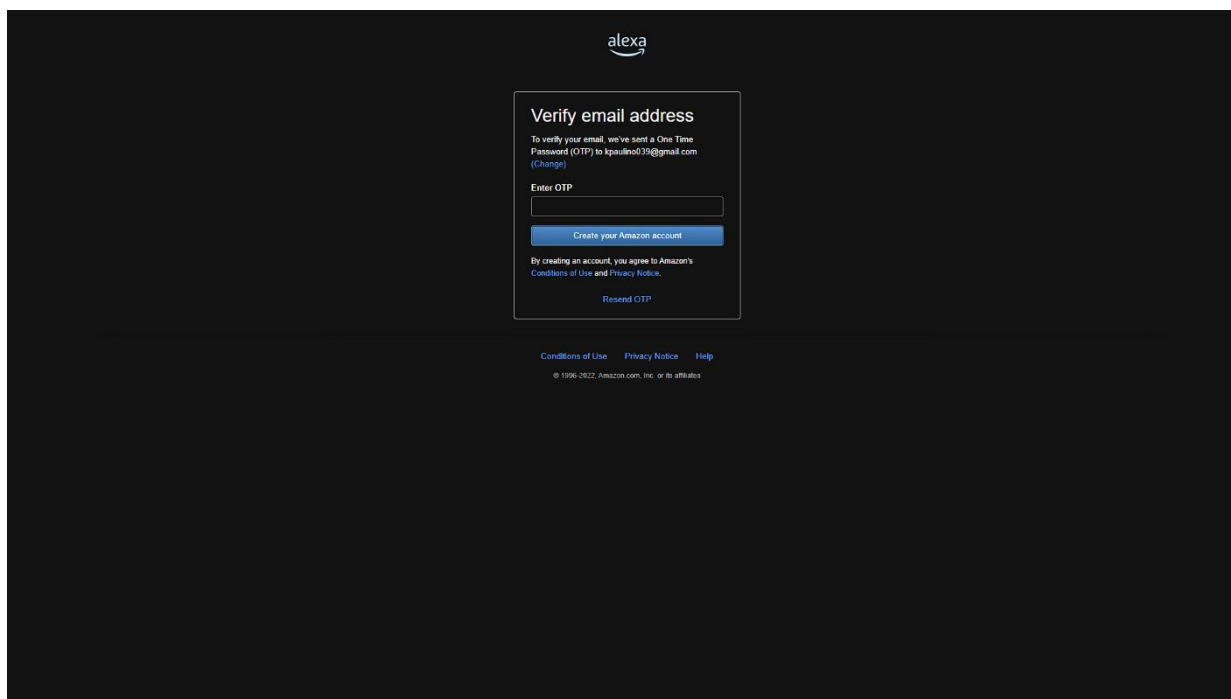
Efetue o cadastro preenchendo todos os campos com seus dados (Nome, E-mail e senha).



The screenshot shows the 'Create account' form for an Amazon Alexa account. The form is centered on a dark background. At the top, the Alexa logo is visible. The form contains the following fields and elements:

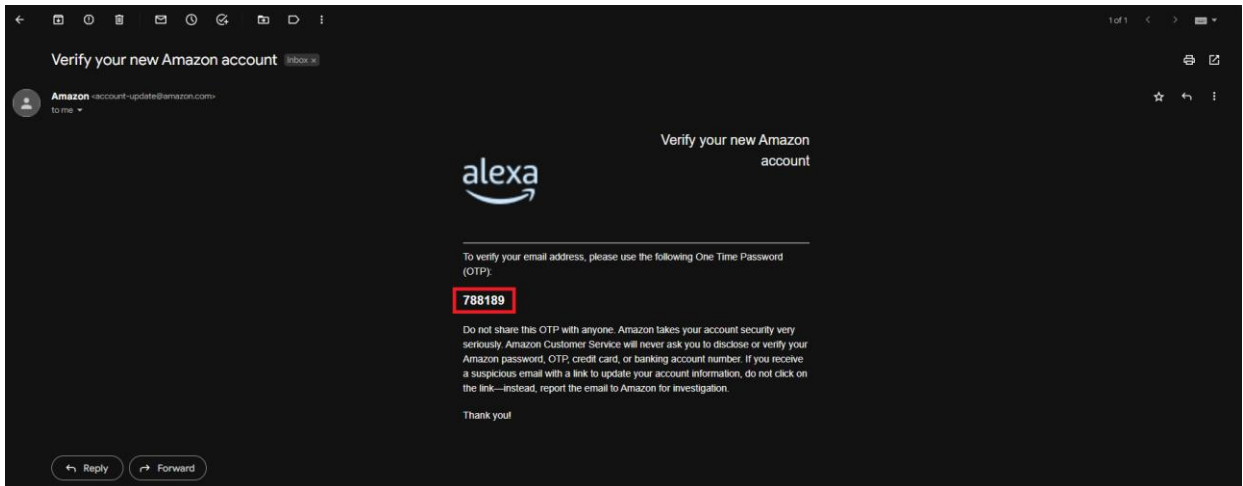
- Your name:** A text input field with the placeholder 'First and last name'.
- Email:** A text input field.
- Password:** A text input field with a strength indicator showing 'At least 6 characters' and a warning 'Passwords must be at least 6 characters'.
- Re-enter password:** A text input field.
- Create your Amazon account:** A blue button.
- By creating an account, you agree to Amazon's Conditions of Use and Privacy Notice.**
- Already have an account? Sign in** (with a right-pointing arrow).
- Footer:** Links for 'Conditions of Use', 'Privacy Notice', and 'Help', followed by the copyright notice '© 1996-2022, Amazon.com, Inc. or its affiliates'.

Após a criação, é necessário inserir um código chamado *OTP* que será enviado para o e-mail cadastrado na seguinte tela:



The screenshot shows the 'Verify email address' form for an Amazon Alexa account. The form is centered on a dark background. At the top, the Alexa logo is visible. The form contains the following elements:

- Verify email address:** The title of the form.
- To verify your email, we've sent a One Time Password (OTP) to kpaullino19@gmail.com (Change):** A message indicating the email address used for verification.
- Enter OTP:** A text input field for the One Time Password.
- Create your Amazon account:** A blue button.
- By creating an account, you agree to Amazon's Conditions of Use and Privacy Notice.**
- Resend OTP:** A link to request a new OTP.
- Footer:** Links for 'Conditions of Use', 'Privacy Notice', and 'Help', followed by the copyright notice '© 1996-2022, Amazon.com, Inc. or its affiliates'.



Após efetuar o cadastro, será necessário completar algumas informações obrigatórias pedidas pelo site:

amazon developer

Search

Amazon Developer Registration

Developer details

1 First name*

2 Last name*

3 Country / Region*
Select your country

4 Developer / Company name*
Enter your developer/company name

5 Phone number*
- CC - Enter your phone number

6 Email address*
Can be different from your login email. Used as primary communication method

Support email address
Enter your customer support email address.

Same as above email address.

The name, email address and phone number entered above won't be displayed in Amazon appstore.

Contact Details

7 Developer / Company address*
Enter your developer/company address

8 City*
Enter your city

9 State*
Enter your state

10 Postal code / Zip code*
Enter postal code

Amazon Developer Services Agreement

I have read and agreed to the terms of the Amazon Developer Services Agreement

Cancel Submit

English

© 2010-2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms Amazon Developer Blog Contact Us

Pode ser usado um nome fictício.
Obs. Recomendo usar um ponto "."

amazon developer

Search

Dashboard Apps & Services Alexa Login with Amazon Amazon Dash Replenishment Reporting Settings

Welcome to Amazon Developer Portal
Tell us about your interests so we can keep you informed about our products and feature updates.

Select the products you're interested in:

Fire TV Amazon Web Services Mobile apps and games
 Twitch Alexa Voice Services Fire Tablets
 Alexa Skills Kit Dash Replenishment Amazon Moments
 Amazon GameOn Software and Video games for PC and MAC

Receive product information and updates

Complete your profile now, or finish it later

Add payment information, user roles and other settings to complete your profile. This can be done later through the Settings tab.

Continue completing your profile

Start exploring the console

Ao concluir o cadastro, você seguirá para a seguinte página:

amazon developer

Search

Dashboard Apps & Services Alexa Login with Amazon Amazon Dash Replenishment Reporting Settings

amazon alexa
Build for voice with Alexa, Amazon's voice service and the brain behind the Amazon Echo

Alexa Skills Kit
A collection of self-service APIs, tools, documentation, and code samples that make it fast and easy for anyone to add skills to Alexa

Alexa Voice Service
Create or manage your Alexa enabled devices

amazon appstore
Build Android apps and games for Amazon Fire TV, Fire tablet, and Amazon's mobile app store

Apps List
View complete list of all your Apps

Reports
View how your app is performing and download reports

Add a New App

amazon dash
Dash Replenishment Console, manage existing devices and set new devices

Device List
View complete list of all your devices

App Settings
View and update your Dash Replenishment app settings

Create a Device

amazon gameon
Engage your players with competitions and leaderboards

Competitions
Create, view and edit competitions

Game Settings
View, modify and add registered games

Add Competitions

amazon
Other services offered for Amazon developers

Login with Amazon
Login with Amazon allows users to login to registered third party websites or apps (clients) using their Amazon user name and password

English

© 2019-2022, Amazon.com, Inc. or its affiliates. All Rights Reserved. Terms Amazon Developer Blog Contact Us

Para acessar a seção de criação de skills:

2. CRIAÇÃO DA SKILL

Logo em seguida ao nosso login, podemos construir nossa skill.

Para iniciar devemos pressionar o botão contendo a seguinte legenda:
Criar skill.

alex developer console

Procurar

Skills Rendimentos Payments Hosting Settings

Alexa Skills Exemplos de skills Learn More

Pesquisar por nome ou ID da skill

Nome da Skill LANGUAGE MODIFICADO STATUS AÇÕES

Criar skill

Alexa Skills

Crie sua primeira skill ou saiba mais sobre Alexa Skills Kit

Criar skill

Ver todas as skills

Recursos

Recommended

Tutorial: How to Build an Engaging Alexa Skill

Guide: The Alexa Skill Design Guide

Blog post: Earn up to 10% commission on eligible product purchases referred from skills by participating in Amazon Associates on Alexa

Popular

Blog Post: 5 tips on how developers can grow revenue from their skills using Alexa Shopping Actions

Video Series: Zero to Hero: A Comprehensive Course to Building an Alexa Skill

Blog post: Como desenvolver e publicar skills de voz com o Amazon Alexa e o AWS

More

Tutorials and code samples

Alexa Design Guide

Webinars

Whitepapers

Alexa LIVE

You've registered for Alexa Live 2022

Join us online on July 20, 2022 at 9:00 AM PST for the keynote

Visit Alexa Live website for more details

Alexa Skill Insights (Beta)

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. Read more in the Alexa Skills Insight Blog.

Lista de tarefas

Make money by enabling in-skill purchases

Learn more

Portuguese (Brasil) © 2010-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved. Termos Documentação Fóruns Blog Alexa Developer Home

Na janela que será aberta devemos nomear a skill no campo **Skill name** como **skill valores** (ou algum de sua preferência), manter o **Primary locale** como **Portuguese (BR)** e permanecer o restante das configurações padrões.

alex developer console

Procurar

Create a new skill

Cancel **Create skill**

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

Skill name

Skill Valores

15/50 characters

Brand names are only allowed if you provide proof of rights in the testing instructions or if you use the brand name in a referential manner that doesn't imply ownership (examples of terms that can be added to a brand name for referential usage: unofficial, unauthorized, fan, fandom, for, about).

Primary locale

A locale refers to a language and the location (country) in which it's spoken. Your primary locale is what you will start building your skill in. You can add locales after your skill is created.

Portuguese (BR)

1. Choose a model to add to your skill

There are many ways to start building a skill. You can design your own custom model or start with a pre-built model. Pre-built models are interaction models that contain a package of intents and utterances that you can add to your skill.

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.

"Alexa, quais são as notícias?"

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.

"Alexa, acenda as luzes da cozinha"

Video

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

"Alexa, reproduza Interstellar"

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

You can provision your own backend resources or you can have Alexa host them for you. If you decide to have Alexa host your skill, you'll get access to our code editor, which will allow you to deploy code directly to AWS Lambda from the developer console.

Alexa-hosted (Node.js) **SELECTED**

Alexa will host skills in your account and get you started with a Node.js template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. [Learn more](#)

Alexa-hosted (Python)

Alexa will host skills in your account and get you started with a Python template. You will gain access to AWS Lambda endpoints in all Alexa service regions, a DynamoDB table for data persistence, and S3 for media storage. [Learn more](#)

Provision your own

Provision your own endpoint and backend resources for your skill. This is recommended for skills that have significant data transfer requirements. You will not gain access to the console's code editor.

Portuguese (Brasil) Feedback X © 2010-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved. Termos Documentação Fóruns Blog Alexa Developer Home

Manter em **Start from Scratch** e **Continue with template**.

alexa developer console

Procure...

Choose a template to add to your skill

Select a skill template from the list below or import a skill shared by the Alexa community as a public Git repository.

Import skill Continue with template

Start from Scratch

This skill gets you started with the required intents and with code demonstrating "Hello World" functionality if you are building an Alexa-hosted skill. [Learn more](#)

By Alexa

Fact Skill

Build an engaging fact skill about any topic. Alexa will select a fact at random and share it with the user when the skill is invoked. [Learn more](#)

Includes: custom intents, Personalization

By Alexa

Scheduling Skill

Build a skill to allow users to schedule appointments on your calendar, receive email confirmations and reminders. [Learn more](#)

Includes: voice permissions, reminders, API calls, session persistence

By Dabble Lab

Survey Skill

Build a stand-up or survey skill that uses passcodes to allow only authorized users to provide updates and respond to questions. [Learn more](#)

Includes: using passcodes, API calls, session persistence

By Dabble Lab

Weather Bot Skill

Build a conversational weather skill to receive a brief weather update for a given location and date. [Learn more](#)

Includes: Alexa Conversations Preview, APL for Audio, session persistence

By Alexa

Portuguese (Brasil) Feedback

© 2019-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved. Termos Documentação Fóruns Blog Alexa Developer Home

alexa developer console

Procure...

Choose a template to add to your skill

Select a skill template from the list below or import a skill shared by the Alexa community as a public Git repository.

Import skill Continue with template

Start from Scratch

This skill gets you started with the required intents and with code demonstrating "Hello World" functionality if you are building an Alexa-hosted skill. [Learn more](#)

By Alexa

Fact Skill

Build an engaging fact skill about any topic. Alexa will select a fact at random and share it with the user when the skill is invoked. [Learn more](#)

Includes: custom intents, Personalization

By Alexa

Scheduling Skill

Build a skill to allow users to schedule appointments on your calendar, receive email confirmations and reminders. [Learn more](#)

Includes: voice permissions, reminders, API calls, session persistence

By Dabble Lab

Survey Skill

Build a stand-up or survey skill that uses passcodes to allow only authorized users to provide updates and respond to questions. [Learn more](#)

Includes: using passcodes, API calls, session persistence

By Dabble Lab

Weather Bot Skill

Build a conversational weather skill to receive a brief weather update for a given location and date. [Learn more](#)

Includes: Alexa Conversations Preview, APL for Audio, session persistence

By Alexa

Creating your Alexa hosted skill.

It will take about a minute.

● ● ●

Provisioning AWS resources...

Portuguese (Brasil) Feedback

© 2019-2022 Amazon.com, Inc. or its affiliates. All Rights Reserved. Termos Documentação Fóruns Blog Alexa Developer Home

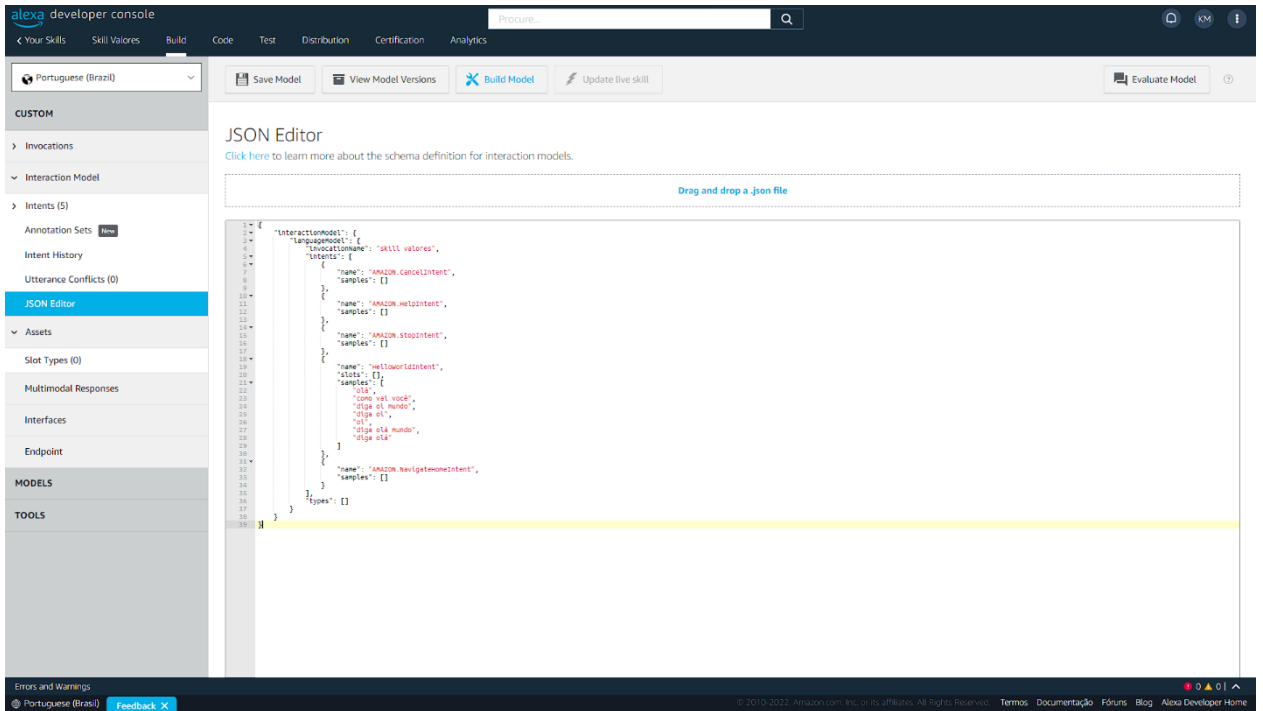
Depois da criação da skill, abra o **Invocation Name** e certificar se no **Skill Invocation Name** está *skill valores*.

The screenshot shows the Alexa Developer Console interface. On the left, there is a navigation menu with sections like 'CUSTOM', 'Invocations', 'Interaction Model', 'Assets', 'Slot Types (0)', 'Multimodal Responses', 'Interfaces', 'Endpoint', 'MODELS', and 'TOOLS'. The main content area is titled 'Alexa Design Guide' and includes a video player for 'Alexa Skills Kit Developer Tutorial for Programmers: Build...'. Below the video, there are 'Resources' links for 'Technical Walkthrough: Toolkit for VS Code' and 'Code First Alexa Skill Development with VS Code and the ASK SDK Controls Framework'. On the right side, there is a 'Skill builder checklist' with four items, each marked as 'REQUIRED' and 'COMPLETE' with a green checkmark:

1. Invocation Name > Enter an invocation name for your skill.
2. Intents, Samples, and Slots > Add at least one intent and one sample utterance.
3. Build Model > Successfully build your interaction model.
4. Endpoint > Set a web service endpoint to handle skill requests.

The screenshot shows the 'Invocation' configuration page in the Alexa Developer Console. The left navigation menu is the same as in the previous screenshot. The main content area is titled 'Invocation' and explains that users say a skill's invocation name to begin an interaction. An example user utterance is shown: 'User: Alexa, ask daily horoscopes for the horoscope for Gemini'. Below this, there is a 'Skill Invocation Name' field with a red box around it, containing the text 'skill valores'. A tooltip for this field says 'How to pick names that are right for you'. Below the field, there is a note in Portuguese: 'Os nomes de marca só são permitidos se você fornecer prova de direitos nas instruções do teste ou se você usar o nome da marca de uma forma referencial que não implique titularidade (exemplos de termos que podem ser adicionados a um nome de marca para uso referencial: não oficial, não autorizado, fã, random, para, sobre). Se o nome de invocação for uma abreviação, deve conter um ponto e um espaço após cada letra (por exemplo: a. b. c.).' At the top of the page, there are buttons for 'Save Model', 'View Model Versions', 'Build Model', 'Update live skill', and 'Evaluate Model'.

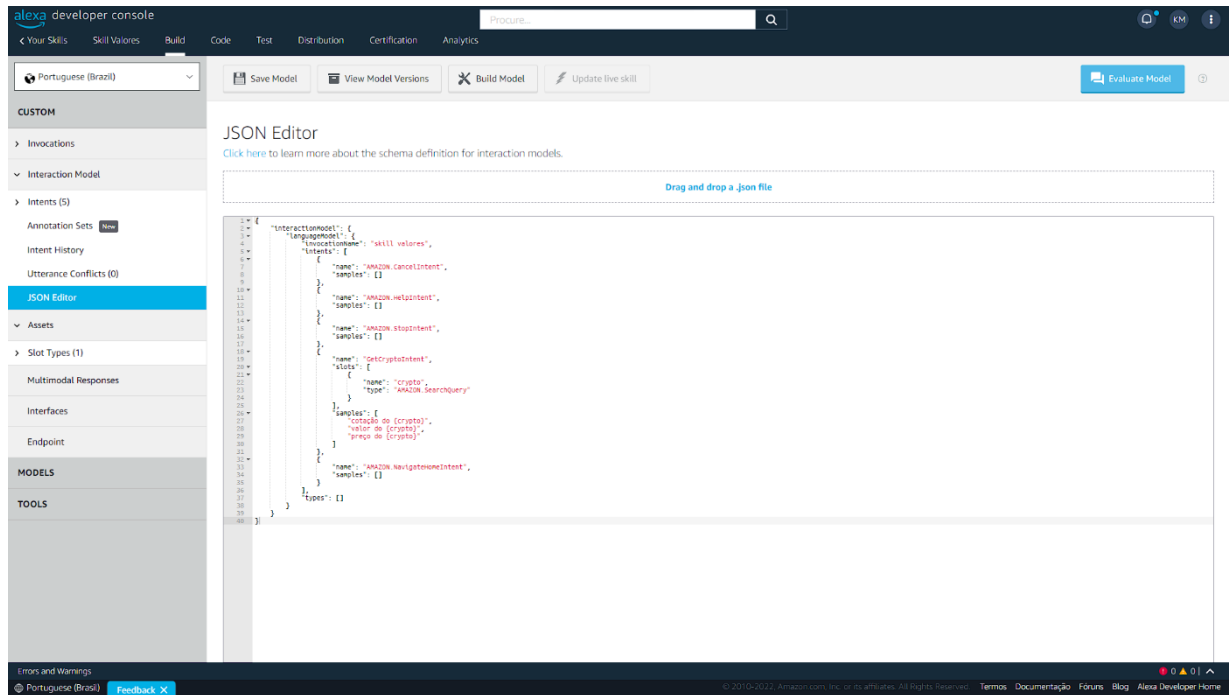
No painel ao lado esquerdo apertar em **Interaction Model** e apertar em **JSON Editor**.



Copiar o seguinte código e colar no editor como mostra o exemplo e logo após salvar em **Save Model**, com o código salvo, clicar em **Build Model**.

```
{
  "interactionModel": {
    "languageModel": {
      "invocationName": "skill valores",
      "intents": [
        {
          "name": "AMAZON.CancelIntent",
          "samples": []
        },
        {
          "name": "AMAZON.HelpIntent",
          "samples": []
        },
        {
          "name": "AMAZON.StopIntent",
          "samples": []
        }
      ]
    }
  }
}
```

```
},  
{  
  "name": "GetCryptoIntent",  
  "slots": [  
    {  
      "name": "crypto",  
      "type": "AMAZON.SearchQuery"  
    }  
  ],  
  "samples": [  
    "cotação do {crypto}",  
    "valor do {crypto}",  
    "preço do {crypto}"  
  ]  
},  
{  
  "name": "AMAZON.NavigateHomeIntent",  
  "samples": []  
}  
],  
"types": []  
}  
},  
"version": "7"  
}
```



Vá para a aba **Code**, copie o seguinte código e cole em **index.js**.

```
/* *
```

** This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v2).*

** Please visit <https://alexa.design/cookbook> for additional examples on implementing slots, dialog management,*

** session persistence, api calls, and more.*

```
*/
```

```
const Alexa = require('ask-sdk-core');
```

```
const LaunchRequestHandler = {
  canHandle(handlerInput) {
    return Alexa.getRequestType(handlerInput.requestEnvelope)
    === 'LaunchRequest';
  },
  handle(handlerInput) {
    const speakOutput = 'Qual criptomoeda deseja cotação?';

    return handlerInput.responseBuilder
      .speak(speakOutput)
```

```

        .reprompt(speakOutput)
        .getResponse();
    }
};

const GetCryptoIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope)
=== 'IntentRequest'
        && Alexa.getIntentName(handlerInput.requestEnvelope) ===
'GetCryptoIntent';
    },
    handle(handlerInput) {

        const crypto = handlerInput.requestEnvelope.request.intent.slots.crypto.value;

        const axios = require('axios');

        return
        axios.get(`https://api.binance.com/api/v3/avgPrice?symbol=${crypto.toUpperCase()}USDT`)
            .then(response => {
                const price = parseFloat(response.data.price).toFixed(2).replace(".", "");

                const speakOutput = `O preço de ${crypto} atualmente é
                ${price}`;

                return handlerInput.responseBuilder
                    .speak(speakOutput)
                    .getResponse();
            })
    }
};

```

```

        .catch(err => {
            const speakOutput = `Houve um erro: ${err.message}`;
            return handlerInput.responseBuilder
                .speak(speakOutput)
                .getResponse();
        })
    }
};

const HelpIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope)
            === 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) ===
            'AMAZON.HelpIntent';
    },
    handle(handlerInput) {
        const speakOutput = 'You can say hello to me! How can I help?';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .reprompt(speakOutput)
            .getResponse();
    }
};

const CancelAndStopIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope)
            === 'IntentRequest'
            && (Alexa.getIntentName(handlerInput.requestEnvelope) ===
            'AMAZON.CancelIntent'

```

```

        // Alexa.getIntentName(handlerInput.requestEnvelope) ==
        'AMAZON.StopIntent');
    },
    handle(handlerInput) {
        const speakOutput = 'Goodbye!';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .getResponse();
    }
};
/**
 * FallbackIntent triggers when a customer says something that
    doesn't map to any intents in your skill
 * It must also be defined in the language model (if the locale supports
    it)
 * This handler can be safely added but will be ignored in locales
    that do not support it yet
 */
const FallbackIntentHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope)
        == 'IntentRequest'
            && Alexa.getIntentName(handlerInput.requestEnvelope) ==
            'AMAZON.FallbackIntent';
    },
    handle(handlerInput) {
        const speakOutput = 'Sorry, I don\'t know about that. Please try
        again.';

        return handlerInput.responseBuilder
            .speak(speakOutput)
            .reprompt(speakOutput)

```

```

        .getResponse();
    }
};
/* *
    * SessionEndedRequest notifies that a session was ended. This
    handler will be triggered when a currently open
    * session is closed for one of the following reasons: 1) The user says
    "exit" or "quit". 2) The user does not
    * respond or says something that does not match an intent defined
    in your voice model. 3) An error occurs
    */

const SessionEndedRequestHandler = {
    canHandle(handlerInput) {
        return Alexa.getRequestType(handlerInput.requestEnvelope)
        === 'SessionEndedRequest';
    },
    handle(handlerInput) {
        console.log(`~~~~~ Session ended:
        ${JSON.stringify(handlerInput.requestEnvelope)}`);
        // Any cleanup logic goes here.
        return handlerInput.responseBuilder.getResponse(); // notice
        we send an empty response
    }
};
/* *
    * The intent reflector is used for interaction model testing and
    debugging.
    * It will simply repeat the intent the user said. You can create custom
    handlers for your intents
    * by defining them above, then also adding them to the request
    handler chain below
    */

const IntentReflectorHandler = {
    canHandle(handlerInput) {

```

```

        return Alexa.getRequestType(handlerInput.requestEnvelope)
=== 'IntentRequest';
    },
    handle(handlerInput) {
        const intentName =
Alexa.getIntentName(handlerInput.requestEnvelope);
        const speakOutput = `You just triggered ${intentName}`;

        return handlerInput.responseBuilder
            .speak(speakOutput)
             //.reprompt('add a reprompt if you want to keep the session
open for the user to respond')
            .getResponse();
    }
};
/**
 * Generic error handling to capture any syntax or routing errors. If
you receive an error
 * stating the request handler chain is not found, you have not
implemented a handler for
 * the intent being invoked or included it in the skill builder below
 */
const ErrorHandler = {
    canHandle() {
        return true;
    },
    handle(handlerInput, error) {
        const speakOutput = 'Sorry, I had trouble doing what you asked.
Please try again.';
        console.log(`~~~~ Error handled: ${JSON.stringify(error)}`);

        return handlerInput.responseBuilder
            .speak(speakOutput)

```



```

    .reprompt(speakOutput)
    .getResponse();
  }
};

exports.handler = Alexa.SkillBuilders.custom()

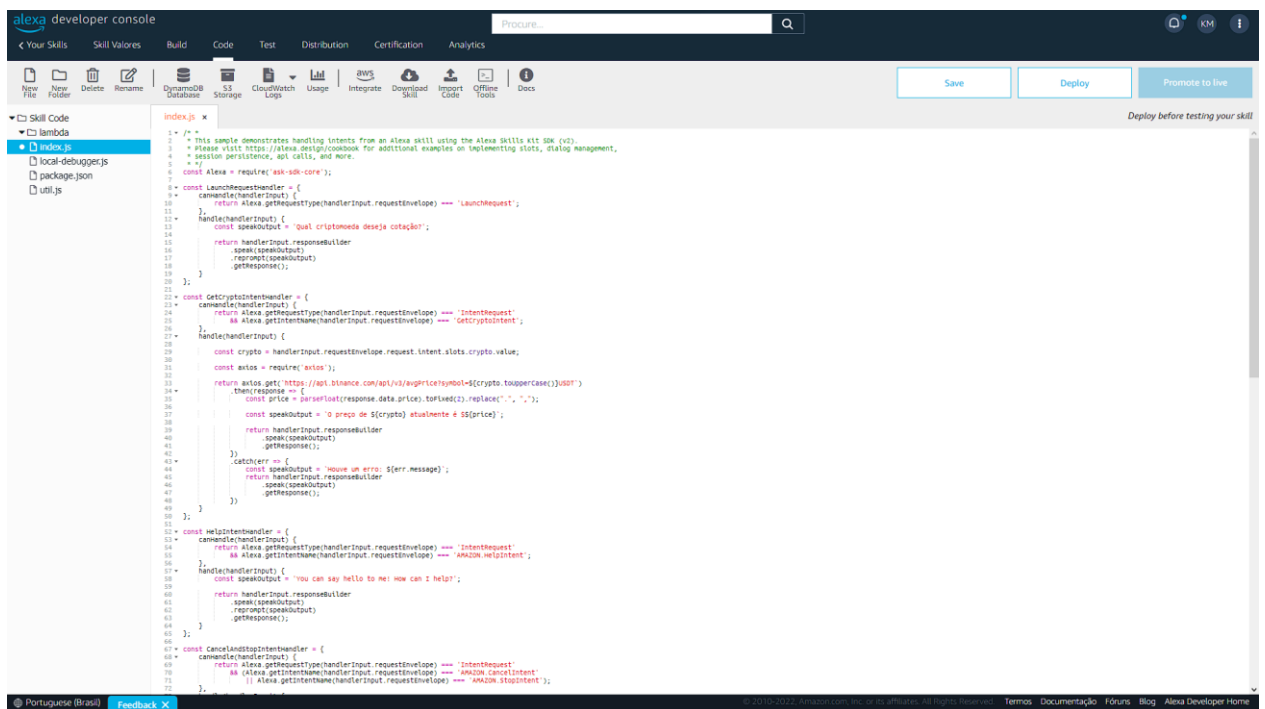
  .addRequestHandlers(
    LaunchRequestHandler,
    GetCryptoIntentHandler,
    HelpIntentHandler,
    CancelAndStopIntentHandler,
    FallbackIntentHandler,
    SessionEndedRequestHandler,
    IntentReflectorHandler)

  .addErrorHandlers(
    ErrorHandler)

  .withCustomUserAgent('sample/hello-world/v1.2')

  .lambda();

```



```

index.js
1  /*
2  * This sample demonstrates handling intents from an Alexa skill using the Alexa Skills Kit SDK (v3).
3  * Please visit https://alexa.design/cookbook for additional examples on implementing slots, dialog management,
4  * session persistence, bot calls, and more.
5  *
6  * @see https://alexa.design/cookbook#use_current_sdk
7  */
8
9  // @ts-check
10 const Alexa = require('ask-sdk-core');
11
12 // Handler for the Launch Intent.
13 const LaunchRequestHandler = {
14   canHandle(handlerInput) {
15     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'LaunchRequest';
16   },
17   handle(handlerInput) {
18     const speakOutput = 'Qual criptomoneda deseja cotação?';
19
20     return handlerInput.responseBuilder
21       .speak(speakOutput)
22       .reprompt(speakOutput)
23       .getResponse();
24   }
25 };
26
27 // Handler for the GetCryptoIntent Intent.
28 const GetCryptoIntentHandler = {
29   canHandle(handlerInput) {
30     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
31       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'GetCryptoIntent';
32   },
33   handle(handlerInput) {
34     const crypto = handlerInput.requestEnvelope.request.intent.slots.crypto.value;
35     const axios = require('axios');
36
37     return axios.get(`https://api.binance.com/api/v3/avgPrice?symbol=${crypto.toUpperCase()}USD`)
38       .then(response => {
39         const price = parseInt(response.data.price, 10).toFixed(2).replace(".", "");
40
41         const speakOutput = `O preço de ${crypto} atualmente é $${price}`;
42
43         return handlerInput.responseBuilder
44           .speak(speakOutput)
45           .getResponse();
46       })
47       .catch(err => {
48         const speakOutput = `Houve um erro: ${err.message}`;
49         return handlerInput.responseBuilder
50           .speak(speakOutput)
51           .getResponse();
52       })
53   }
54 };
55
56 // Handler for the Help Intent.
57 const HelpIntentHandler = {
58   canHandle(handlerInput) {
59     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
60       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.HelpIntent';
61   },
62   handle(handlerInput) {
63     const speakOutput = 'You can say hello to me! How can I help?';
64
65     return handlerInput.responseBuilder
66       .speak(speakOutput)
67       .reprompt(speakOutput)
68       .getResponse();
69   }
70 };
71
72 // Handler for the Cancel and Stop Intent.
73 const CancelAndStopIntentHandler = {
74   canHandle(handlerInput) {
75     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
76       && (Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.CancelIntent'
77         || Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.StopIntent');
78   },
79   handle(handlerInput) {
80     const speakOutput = 'Goodbye!';
81
82     return handlerInput.responseBuilder
83       .speak(speakOutput)
84       .getResponse();
85   }
86 };
87
88 // Handler for the Fallback Intent.
89 const FallbackIntentHandler = {
90   canHandle(handlerInput) {
91     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'IntentRequest'
92       && Alexa.getIntentName(handlerInput.requestEnvelope) === 'AMAZON.FallbackIntent';
93   },
94   handle(handlerInput) {
95     const speakOutput = 'Sorry, I did not understand that. Please try again.';
96
97     return handlerInput.responseBuilder
98       .speak(speakOutput)
99       .reprompt(speakOutput)
100      .getResponse();
101   }
102 };
103
104 // Handler for the SessionEnded Request.
105 const SessionEndedRequestHandler = {
106   canHandle(handlerInput) {
107     return Alexa.getRequestType(handlerInput.requestEnvelope) === 'SessionEndedRequest';
108   },
109   handle(handlerInput) {
110     const speakOutput = 'Thank you for using the skill. Goodbye!';
111
112     return handlerInput.responseBuilder
113       .speak(speakOutput)
114       .getResponse();
115   }
116 };
117
118 // Handler for the Unhandled Request.
119 const UnhandledRequestHandler = {
120   handle(handlerInput) {
121     const speakOutput = 'Sorry, I did not understand that. Please try again.';
122
123     return handlerInput.responseBuilder
124       .speak(speakOutput)
125       .reprompt(speakOutput)
126       .getResponse();
127   }
128 };
129
130 // The lambda function handler.
131 exports.handler = Alexa.SkillBuilders.custom()
132   .addRequestHandlers(
133     LaunchRequestHandler,
134     GetCryptoIntentHandler,
135     HelpIntentHandler,
136     CancelAndStopIntentHandler,
137     FallbackIntentHandler,
138     SessionEndedRequestHandler,
139     IntentReflectorHandler)
140   .addErrorHandlers(
141     ErrorHandler)
142   .withCustomUserAgent('sample/hello-world/v1.2')
143   .lambda();

```

Copie o seguinte código e cole em **local-debugger.js**.

*/**

** Copyright 2019 Amazon.com, Inc. or its affiliates. All Rights Reserved.*

** Licensed under the Apache License, Version 2.0 (the "License").*

** You may not use this file except in compliance with the License.*

** A copy of the License is located at*

** <http://www.apache.org/licenses/LICENSE-2.0>*

** or in the "license" file accompanying this file. This file is distributed*

** on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either*

** express or implied. See the License for the specific language governing*

** permissions and limitations under the License.*

**/*

/ ## DEPRECATION NOTICE*

This script has been deprecated and is no longer supported.

Please use the [ASK Toolkit for VS Code]

(<https://marketplace.visualstudio.com/items?itemName=ask-toolkit.alex-skill-kit-toolkit>),

which provides a more end-to-end integration with Visual Studio Code. If you

use another editor/IDE, please check out the [ASK SDK Local Debug package at npm]

(<https://www.npmjs.com/package/ask-sdk-local-debug>).

**/*

const net = require('net');

const fs = require('fs');

```

const localDebugger = net.createServer();

const httpHeaderDelimiter = '\r\n';
const httpBodyDelimiter = '\r\n\r\n';
const defaultHandlerName = 'handler';
const host = 'localhost';
const defaultPort = 0;

/**
 * Resolves the skill invoker class dependency from the user
provided
 * skill entry file.
 */

// eslint-disable-next-line import/no-dynamic-require
const skillInvoker = require(getAndValidateSkillInvokerFile());
const portNumber = getAndValidatePortNumber();
const lambdaHandlerName = getLambdaHandlerName();

/**
 * Starts listening on the port for incoming skill requests.
 */

localDebugger.listen(portNumber, host, () => {
  console.log(`Starting server on port:
${localDebugger.address().port}`);
});

/**
 * For a new incoming skill request a new socket connection is
established.

```

** From the data received on the socket the request body is extracted, parsed into*

** JSON and passed to the skill invoker's lambda handler.*

** The response from the lambda handler is parsed as a HTTP 200 message format as specified*

** here - <https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#http-header-1>*

** The response is written onto the socket connection.*

**/*

```

localDebugger.on('connection', (socket) => {
    console.log(`Connection from:
    ${socket.remoteAddress}:${socket.remotePort}`);
    socket.on('data', (data) => {
        const body =
        JSON.parse(data.toString().split(httpBodyDelimiter).pop());
        console.log(`Request envelope: ${JSON.stringify(body)}`);
        skillInvoker[lambdaHandlerName](body, null, (_invokeErr,
        response) => {
            response = JSON.stringify(response);
            console.log(`Response envelope: ${response}`);
            socket.write(`HTTP/1.1 200
            OK${httpHeaderDelimiter}Content-Type: application/json;charset=UTF-
            8${httpHeaderDelimiter}Content-Length:
            ${response.length}${httpBodyDelimiter}${response}`);
        });
    });
});

/**
 * Validates user specified port number is in legal range [0, 65535].
 * Defaults to 0.
 */

```

```

function getAndValidatePortNumber() {
    const portNumberArgument =
Number(getArgument('portNumber', defaultPort));
    if (!Number.isInteger(portNumberArgument)) {
        throw new Error(`Port number has to be an integer -
${portNumberArgument}.`);
    }
    if (portNumberArgument < 0 || portNumberArgument > 65535) {
        throw new Error(`Port out of legal range:
${portNumberArgument}. The port number should be in the range [0,
65535]`);
    }
    if (portNumberArgument === 0) {
        console.log('The TCP server will listen on a port that is free.'
+ 'Check logs to find out what port number is being used');
    }
    return portNumberArgument;
}

/**
 * Gets the lambda handler name.
 * Defaults to "handler".
 */

function getLambdaHandlerName() {
    return getArgument('lambdaHandler', defaultHandlerName);
}

/**
 * Validates that the skill entry file exists on the path specified.
 * This is a required field.
 */

```

```

// eslint-disable-next-line consistent-return
function getAndValidateSkillInvokerFile() {
  const fileNameArgument = getArgument('skillEntryFile');
  if (!fs.existsSync(fileNameArgument)) {
    throw new Error(`File not found: ${fileNameArgument}`);
  }
  return fileNameArgument;
}

/**
 * Helper function to fetch the value for a given argument
 * @param {argumentName} argumentName name of the argument
for which the value needs to be fetched
 * @param {defaultValue} defaultValue default value of the argument
that is returned if the value doesn't exist
 */

function getArgument(argumentName, defaultValue) {
  const index = process.argv.indexOf(`--${argumentName}`);
  if (index === -1 || typeof process.argv[index + 1] === 'undefined') {
    if (defaultValue === undefined) {
      throw new Error(`Required argument - ${argumentName} not
provided.`);
    } else {
      return defaultValue;
    }
  }
  return process.argv[index + 1];
}

```

```

1  /*
2  * Copyright 2019 Amazon.com, Inc. or its affiliates. All rights reserved.
3  * Licensed under the Apache License, Version 2.0 (the "License");
4  * you may not use this file except in compliance with the License.
5  * A copy of the License is located at
6  * http://www.apache.org/licenses/LICENSE-2.0
7  *
8  * or in the "license" file accompanying this file. This file is distributed
9  * on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
10 * express or implied. See the License for the specific language governing
11 * permissions and limitations under the License.
12 */
13
14 /** DEPRECATION NOTICE
15
16 This script has been deprecated and is no longer supported.
17 Please use the alexa-toolkit-for-js code
18 https://aws.amazon.com/blogs/aws/2019-08-28-alexa-toolkit-alexa-skills-kit-toolkit/,
19 which provides a more end-to-end experience with Visual Studio Code, if you
20 use another editor/IDE, please check out the alexa-local-debugger package at npm
21 (https://www.npmjs.com/package/alexa-skills-kit-local-debugger).
22 */
23
24
25
26 const net = require('net');
27 const fs = require('fs');
28
29 const localDebugger = net.createServer();
30
31 const httpHeaderLine = '\r\n';
32 const httpHeaderName = 'handler';
33 const defaultHandler = 'handler';
34 const host = 'localhost';
35 const defaultPort = 80;
36
37 /**
38 * Resolves the skill handler class dependency from the user provided
39 * skill entry file.
40 */
41 // eslint-disable-next-line import/no-dynamic-require
42 const skillHandler = require(getAbsolutePath(skillHandler));
43 const portNumber = getAndValidatePortNumber();
44 const lambdaHandlerName = getLambdaHandlerName();
45
46 /**
47 * Starts listening on the port for incoming skill requests.
48 */
49 localDebugger.listen(portNumber, host, () => {
50   console.log('Starting server on port: ' + localDebugger.address().port);
51 });
52
53 /**
54 * For a new incoming skill request a new socket connection is established.
55 * From the data received on the socket the request body is extracted, parsed into
56 * JSON and passed to the skill handler & lambda handler.
57 * The response from the lambda handler is parsed as a HTTP message format as specified
58 * here: https://developer.amazon.com/docs/custom-skills/request-and-response-json-reference.html#http-header-1
59 * The response is written onto the socket connection.
60 */
61 localDebugger.on('connection', (socket) => {
62   console.log('connection from: ' + socket.remoteAddress + ':' + socket.remotePort);
63   socket.on('data', (data) => {
64     const body = JSON.parse(data.toString()).split(httpHeaderLine).pop();
65     console.log('request envelope: ' + JSON.stringify(body));
66     skillHandler(lambdaHandlerName)(body, null, { 'handler': response }) => {
67       response = JSON.stringify(response);
68       console.log('response envelope: ' + response);
69       socket.write('HTTP/1.1 200 OK ' + localDebugger.address().port + '\r\n' + httpHeaderLine + response);
70     }
71   });
72 });

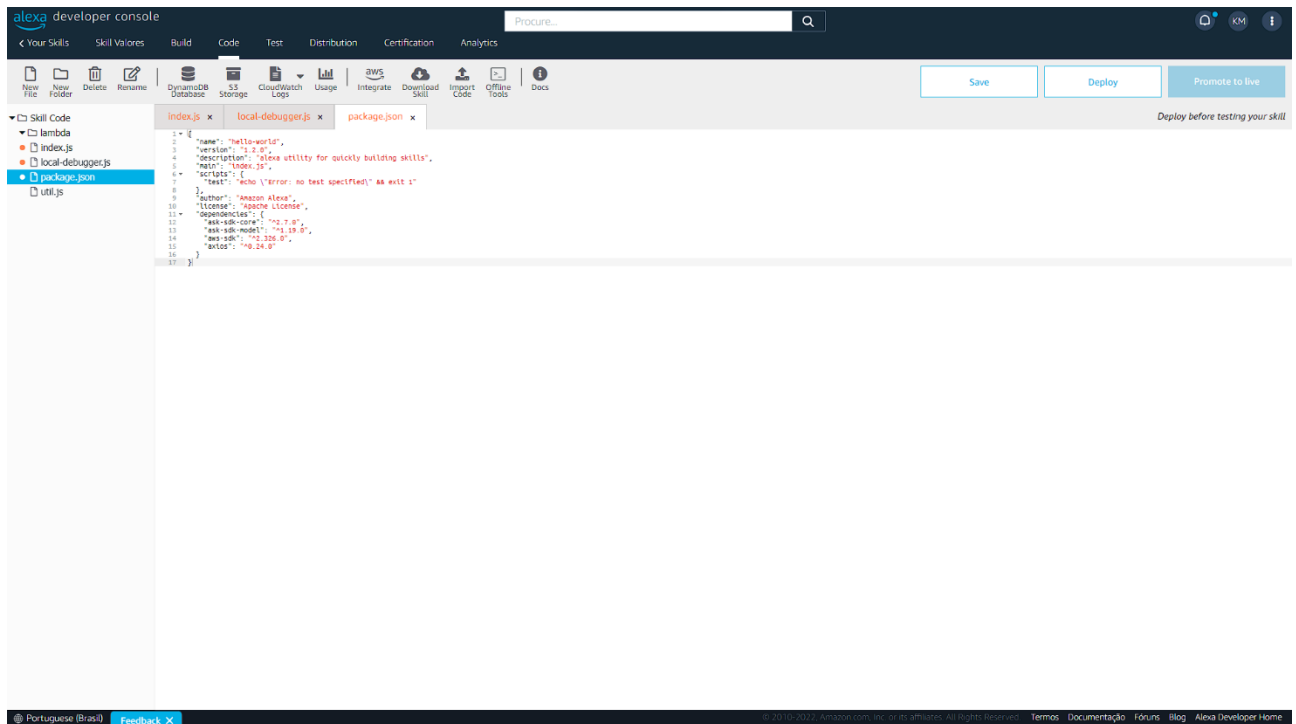
```

Copie o seguinte código e cole em **package.json**.

```

{
  "name": "hello-world",
  "version": "1.2.0",
  "description": "alexa utility for quickly building skills",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "Amazon Alexa",
  "license": "Apache License",
  "dependencies": {
    "ask-sdk-core": "^2.7.0",
    "ask-sdk-model": "^1.19.0",
    "aws-sdk": "^2.326.0",
    "axios": "^0.24.0"
  }
}

```



Copie o seguinte código e cole em `util.js`.

```
const AWS = require('aws-sdk');
```

```
const s3SigV4Client = new AWS.S3({
  signatureVersion: 'v4',
  region: process.env.S3_PERSISTENCE_REGION
});
```

```
module.exports.getS3PreSignedUrl = function
getS3PreSignedUrl(s3ObjectKey) {
```

```
  const bucketName = process.env.S3_PERSISTENCE_BUCKET;
  const s3PreSignedUrl = s3SigV4Client.getSignedUrl('getObject', {
    Bucket: bucketName,
    Key: s3ObjectKey,
    Expires: 60*1 // the Expires is capped for 1 minute
  });
```

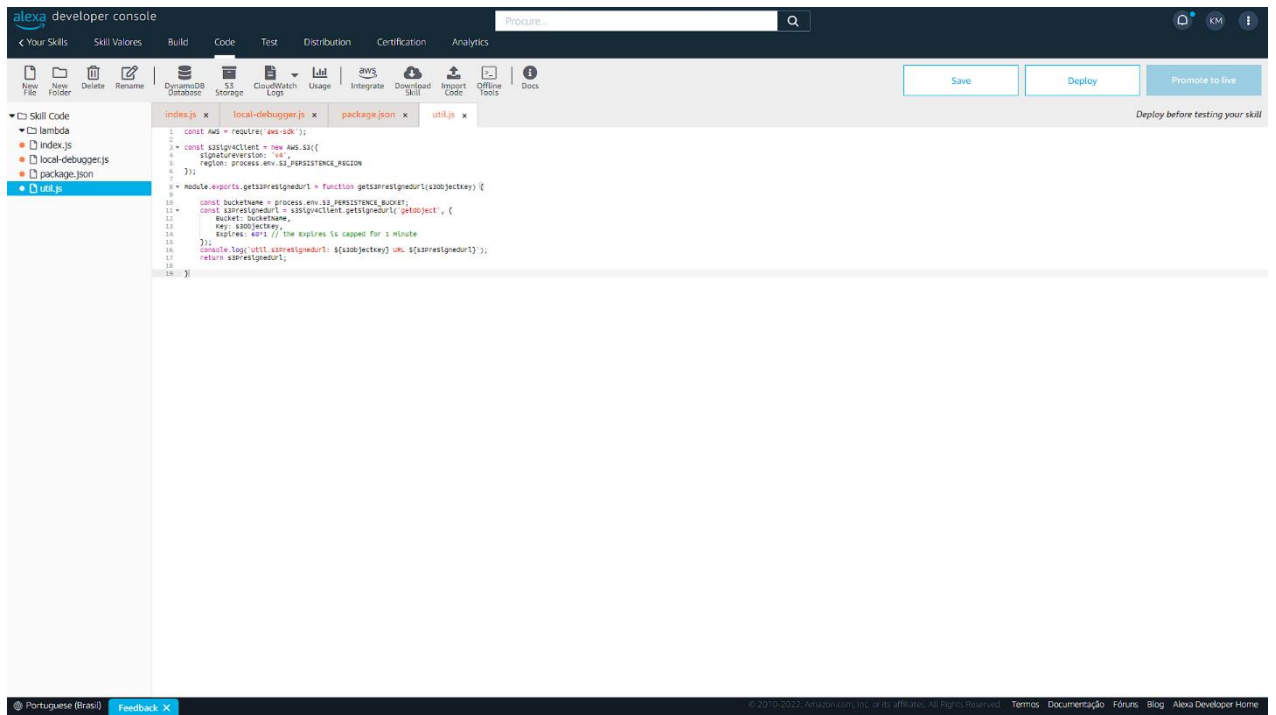


```

console.log(`Util.s3PreSignedUrl:    ${s3ObjectKey}    URL
${s3PreSignedUrl}`);

return s3PreSignedUrl;
}

```

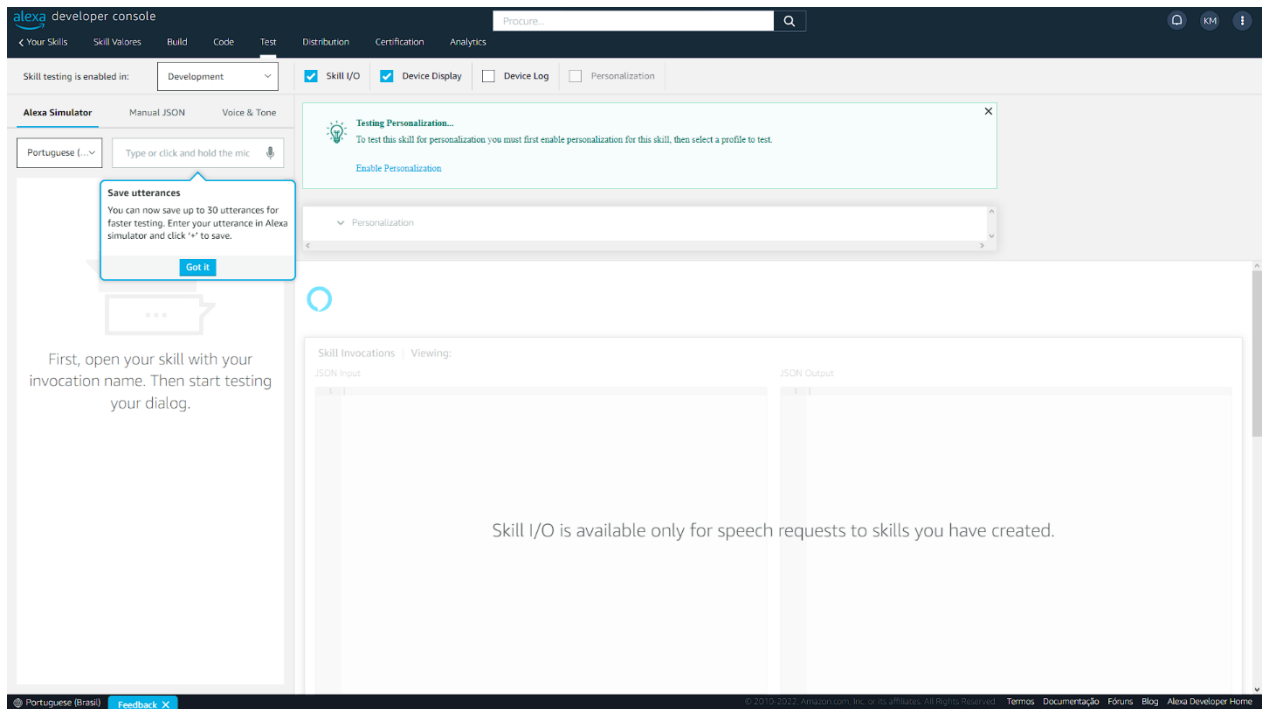


Após todos códigos inseridos salve todos em cada aba no botão **Save** e logo após **Deploy**.

Após isso a skill já pode ser testada acessando a aba **Test**.

The screenshot shows the Alexa Developer Console interface. At the top, the navigation bar includes 'Your Skills', 'Skill Values', 'Build', 'Code', 'Test', 'Distribution', 'Certification', and 'Analytics'. The 'Test' tab is selected. A notification at the top states 'Test is disabled for this skill.' with a dropdown menu set to 'Off'. Below this, a message reads: 'When test is enabled, you can interact with the selected stage of your skill in the Alexa simulator and on all devices linked to your Alexa developer account'. The main content area features a 'Type or click and hold the mic' input field, a 'Save utterances' tooltip, and a 'Skill I/O' section. The 'Skill I/O' section contains a message: 'Skill I/O is available only for speech requests to skills you have created.' The footer includes 'Portuguese (Brasil)', a 'Feedback' button, and copyright information for Amazon.com, Inc. or its affiliates.

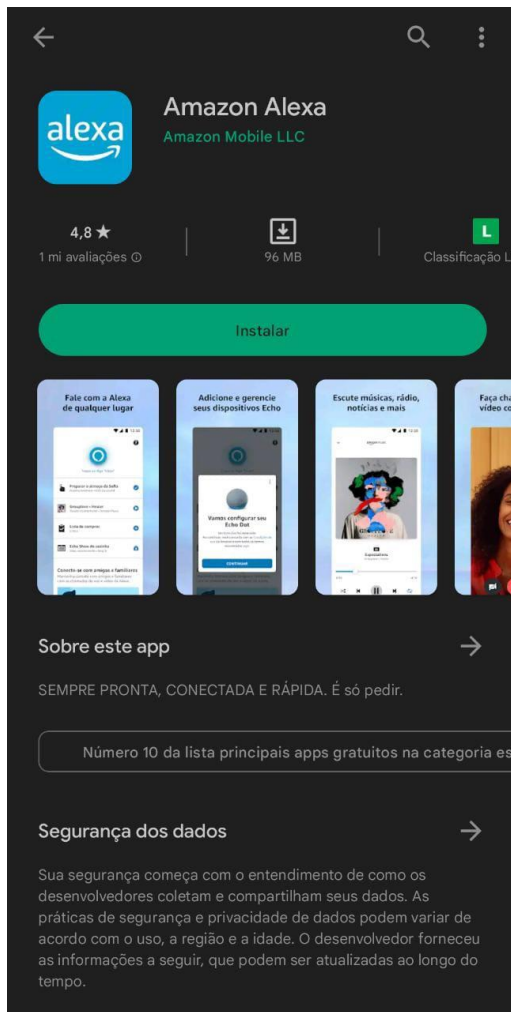
This screenshot is identical to the one above, but the 'Test' dropdown menu is now set to 'Development'. The rest of the interface, including the 'Skill I/O' message and footer, remains the same.



3. UTILIZANDO NA ALEXA

Caso tenha uma Alexa em casa e queira testar a *skill* nela siga as seguintes instruções: *(Vale lembrar que o e-mail cadastrado no console tem que ser o mesmo que usado na sua Alexa.)*

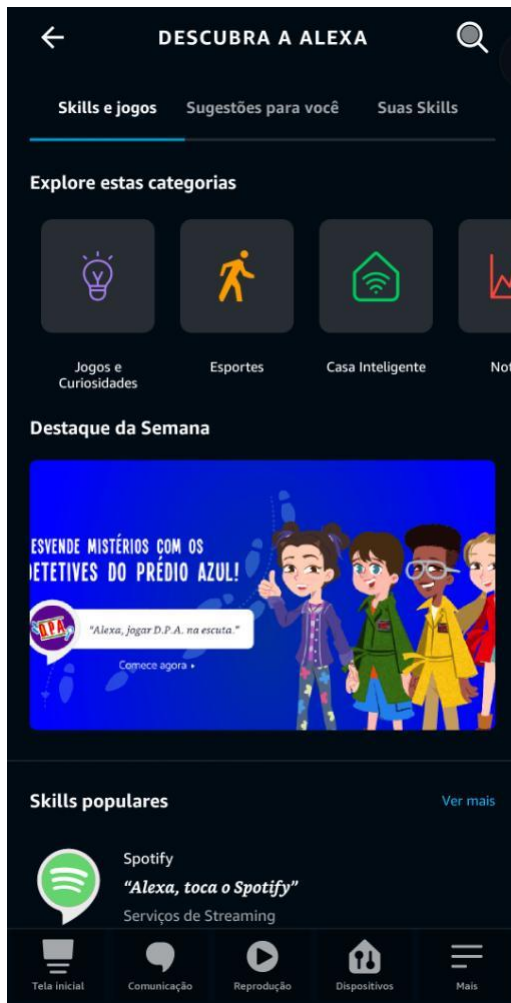
Primeiramente abra o aplicativo **Amazon Alexa**.



Siga para a aba **Mais**.



Aperte em **Skills e jogos**, logo após em **Suas skills**.



desenv. E aperte na **skill** valores.



E aperte em **ATIVAR PARA O USO**.