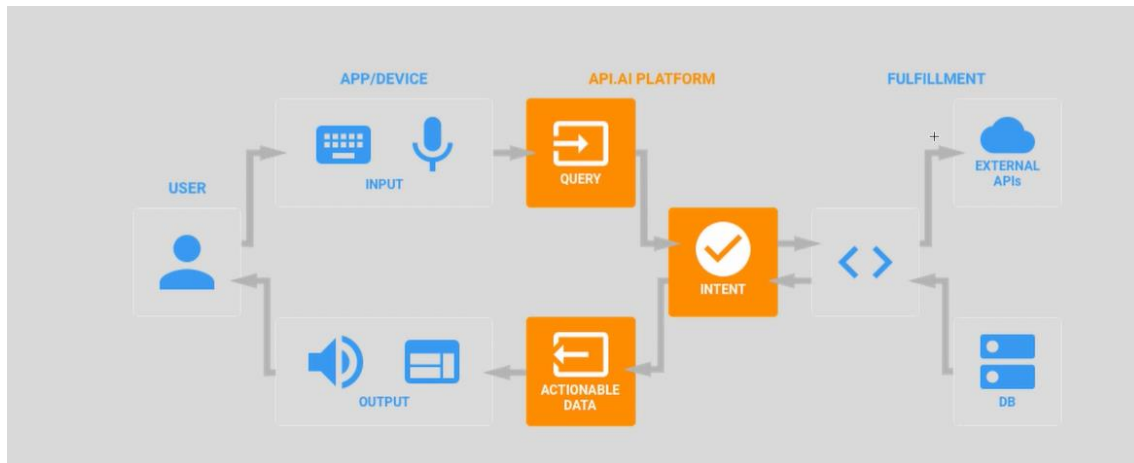


## Plano de aula 06 – WebHook

Nós trabalhamos até então, com um usuário fazendo uma consulta ao agente. O agente analisava uma entrada de texto ou voz e respondia de volta ao usuário via texto ou via voz. Mais potencialmente nós podemos conectar essa intente a um WebHook onde o código será executado no nosso servidor para retornar a resposta usuário.



Vamos criar uma intente para consultar preço de curso, ainda usando uma resposta padrão, sem usar Webhook.

- 1) Crie uma entidade chamada curso
- 2) Entre com os seguintes cursos:
  - a. Python
  - b. R
  - c. Machine Learning → ML
  - d. Hadoop
- 3) Crie uma intente chamada preco.curso
- 4) Entre com as seguintes frases para treinamento
  - a. Qual o preço do curso de Machine Learning?
  - b. Qual o valor do curso de R?
  - c. Quanto custa o curso @curso?
- 5) Resposta: 100 reais.
- 6) Salve e teste

A partir desse ponto vamos trabalhar com WebHook para retornar uma resposta do servidor.

O código que será executado em um backend pode estar em um servidor da empresa, pode estar em uma plataforma de cloud computing, ele pode estar em uma máquina com o sistema operacional windows e linguagem C#, ele pode estar em uma máquina que usa linux com php e assim por diante.

- 1) Vá até a opção fulfillment e habilite a opção webhook.
- 2) Observe a Figura 1.

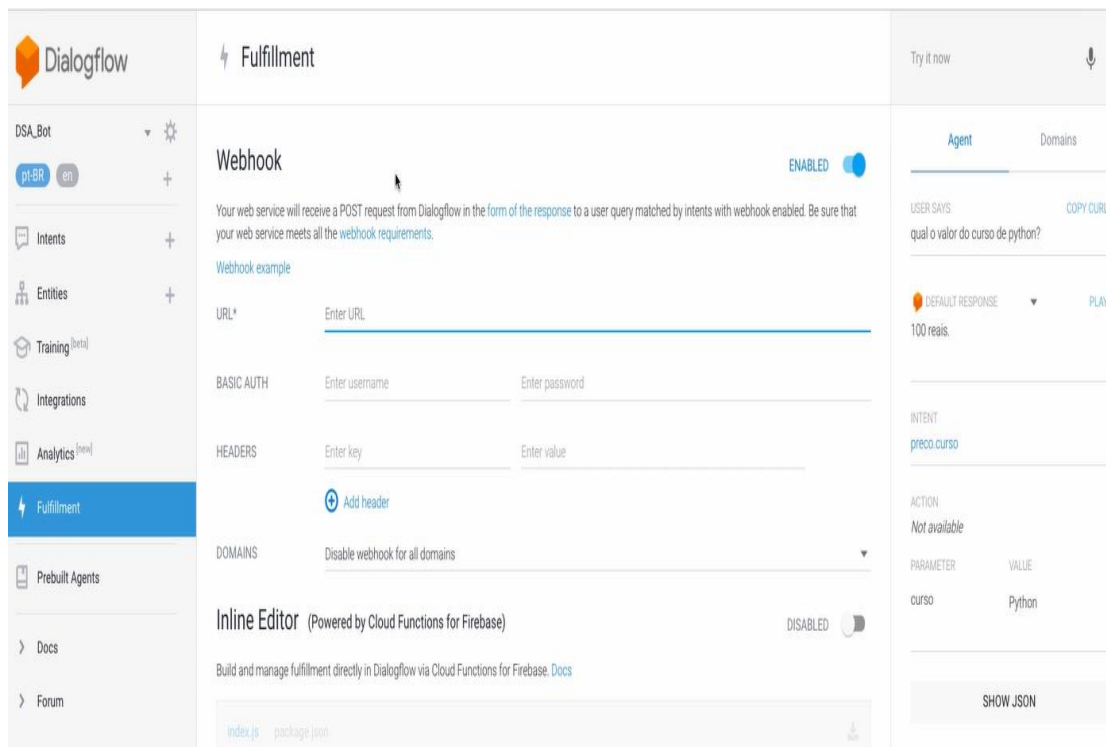


Figura 1

Não é obrigado você trabalhar com WebHook em todas as intentes. Você pode configurar individualmente cada intente que usará WebHook.

Observe que você pode informar uma URL, um login e uma senha para seu agente.

- 3) Habilite Inline Editor e observe que ele fornece dois arquivos index.js e package.json. veja a Figura 2.



Figura 2

A qualquer momento você pode fazer download do arquivo, clicando na seta do lado direito superior.

- 4) Faça o deploy do index.js clicando no botão DEPLY, como mostra a Figura 3.

Isso envia o código para o serviço gerenciado Google Cloud Function que irá executar esse código quando Dialogflow enviar uma requisição a ele.

Agora vamos configurar uma intente para utilizar WebHook.

- 1) Quando fizemos o deploy na seção anterior, nós conectamos nosso código index.js ao Google Cloud Functions.
- 2) Abra a intente que você quer usar WebHook e habilite o seu uso, como mostra a Figura 4.

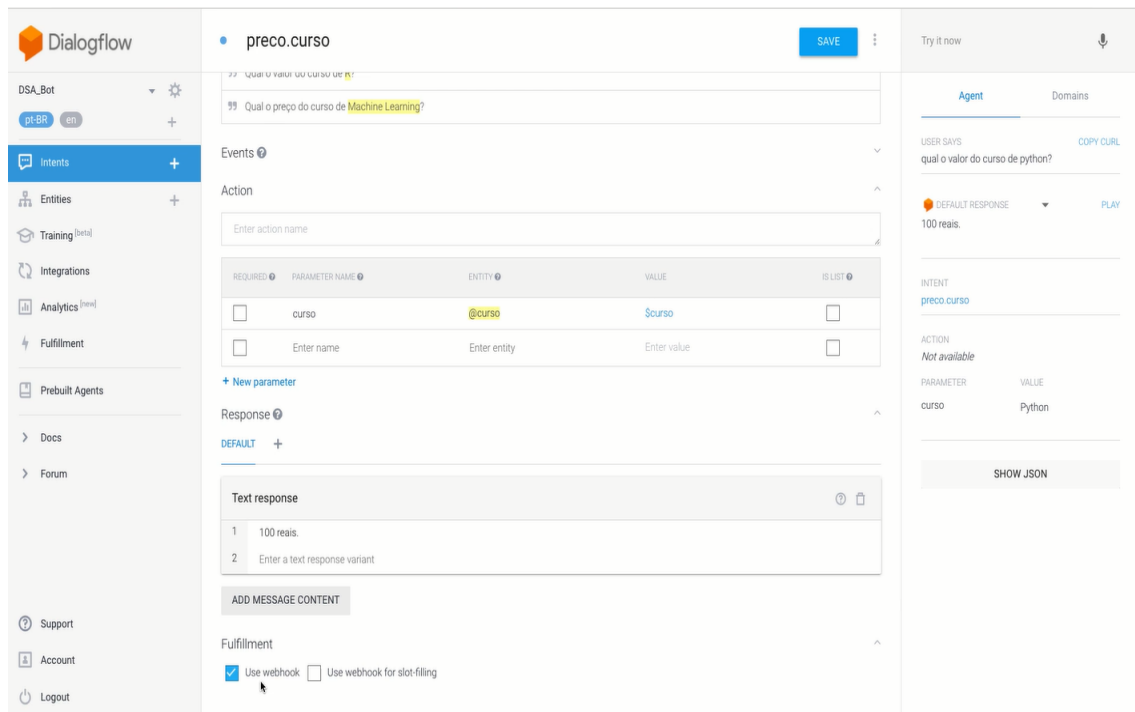


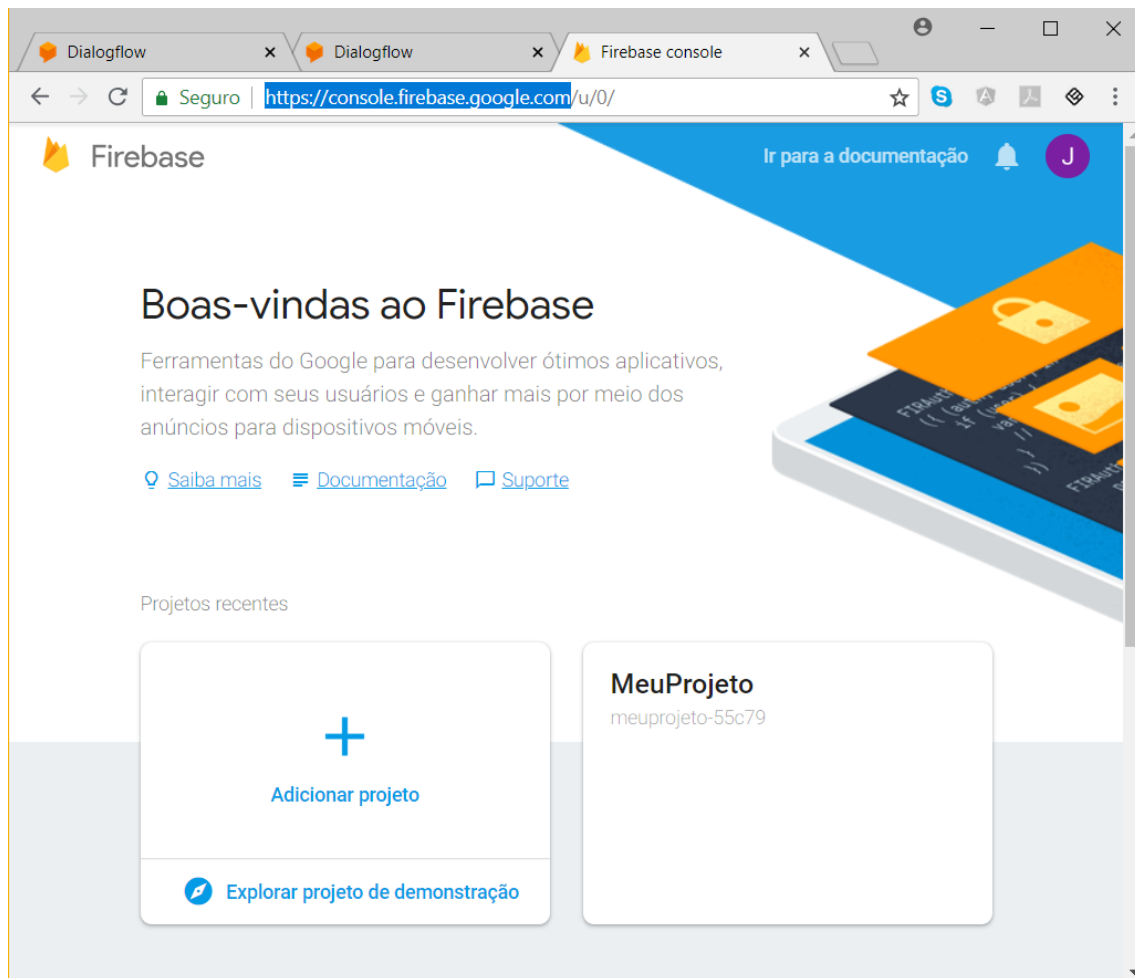
Figura 4. Habilitando a intenção para usar WebHook.

- 3) Sete uma action na intente preco.curso com a string “**preco.curso**”. Não é obrigado ser o mesmo nome da intente pode ser outro. Uma action é uma string que será passada para o servidor.

## A plataforma Firebase

Esse a plataforma Firebase, usando a seguinte URL: <https://console.firebase.google.com>

Você será direcionado para a seguinte tela

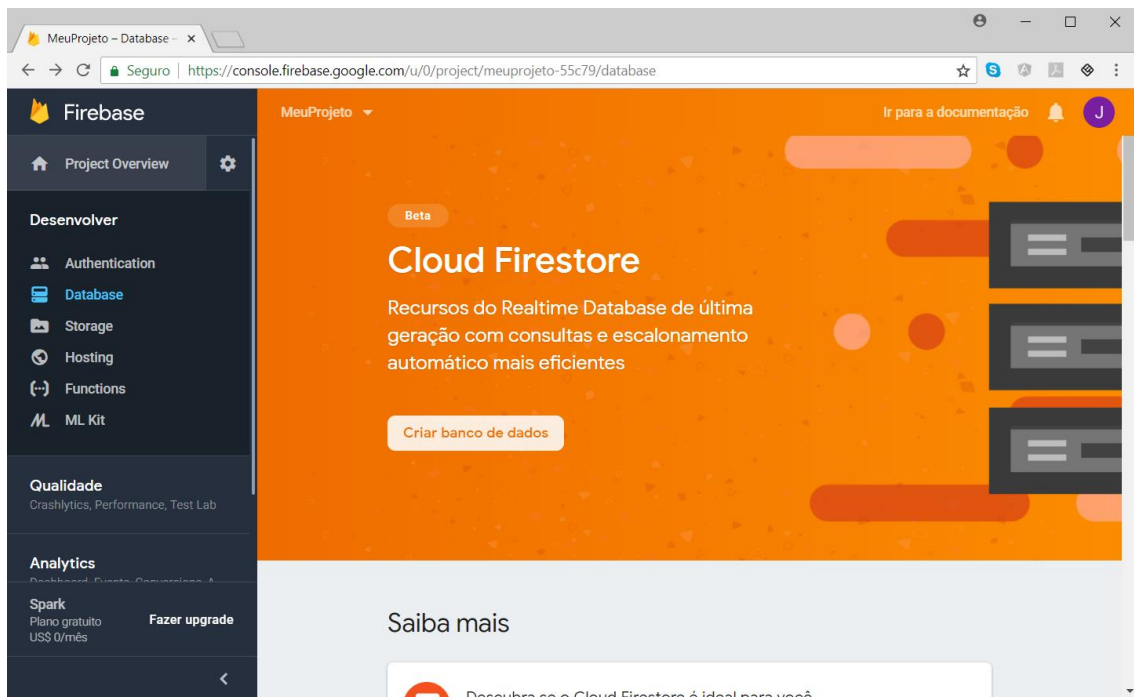


Se você já estiver conectado ao Google, com o login que você está desenvolvendo seu bot, o mesmo irá aparecer na tela. Então, dê dois cliques no mesmo para acessá-lo. Você será direcionado para a tela a seguir:

### Google Cloud Firestore

Volte para o console de seu projeto.

Clique na opção **Database** e você será direcionado para a tela



Nessa tela temos duas opções de banco de dados: **Cloud Firestore e Realtime Database**

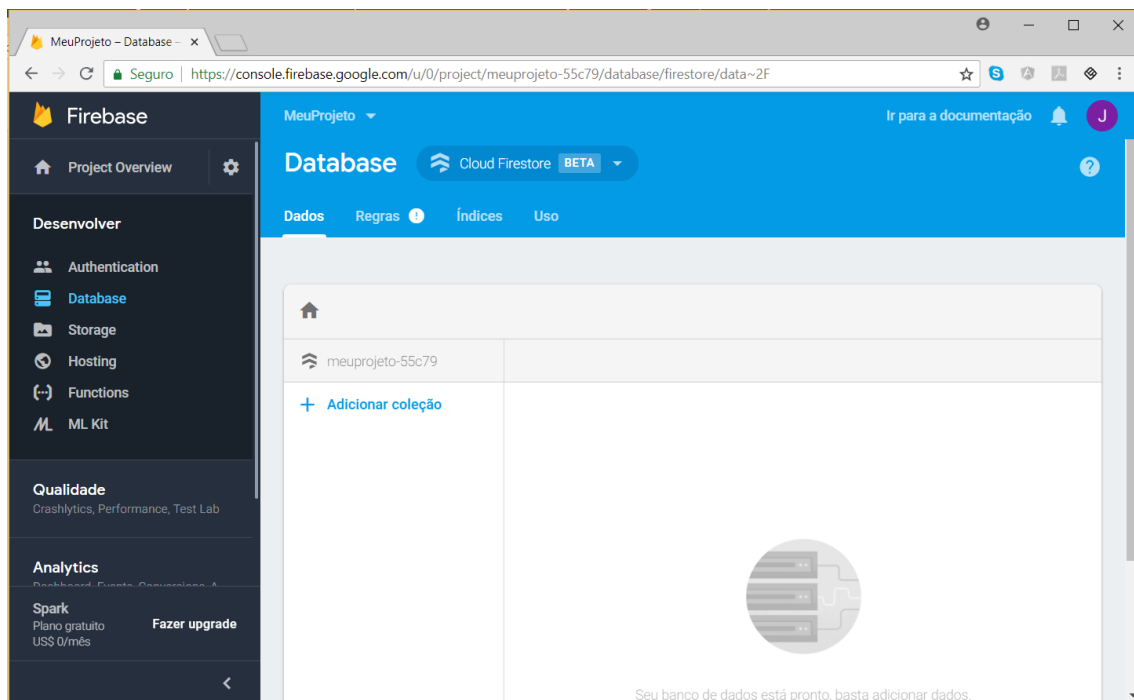
Para esse projeto de teste, vamos trabalhar com a opção Cloud Firestore. Portanto, clique em Criar banco de dados e, você irá para a seguinte tela



Nessa tela temos duas opções modo bloqueado e modo de teste. Vamos selecionar a opção modo de teste, por enquanto. Veja o aviso que o sistema fornece



Clique em ativar e você será levado para a seguinte tela



Nessa janela você crie as suas coleções, define as regras de acesso e assim por diante.

### Conceitos de Cloud Firestore

O **Cloud Fierstore** é um banco de dados escalável que roda em uma plataforma de nuvem e ele não é um banco de dados relacionais. Ele tem alguns conceitos importantes como: documentos que serão criados dentro de coleções, como por exemplo nós podemos ter uma coleção de

curso e os documentos nesta coleção, sendo que, cada documento corresponde a cursos diferentes.

### Criando coleções e documentos no Cloud Firestore

Crie a coleção cursos com os documentos

#### R

Campos: nome=R fundamentos, valor=250 e instrutor= José Cunha

#### Python

Campos: nome=Introdução ao Python, valor=300 e instrutor= Pedro

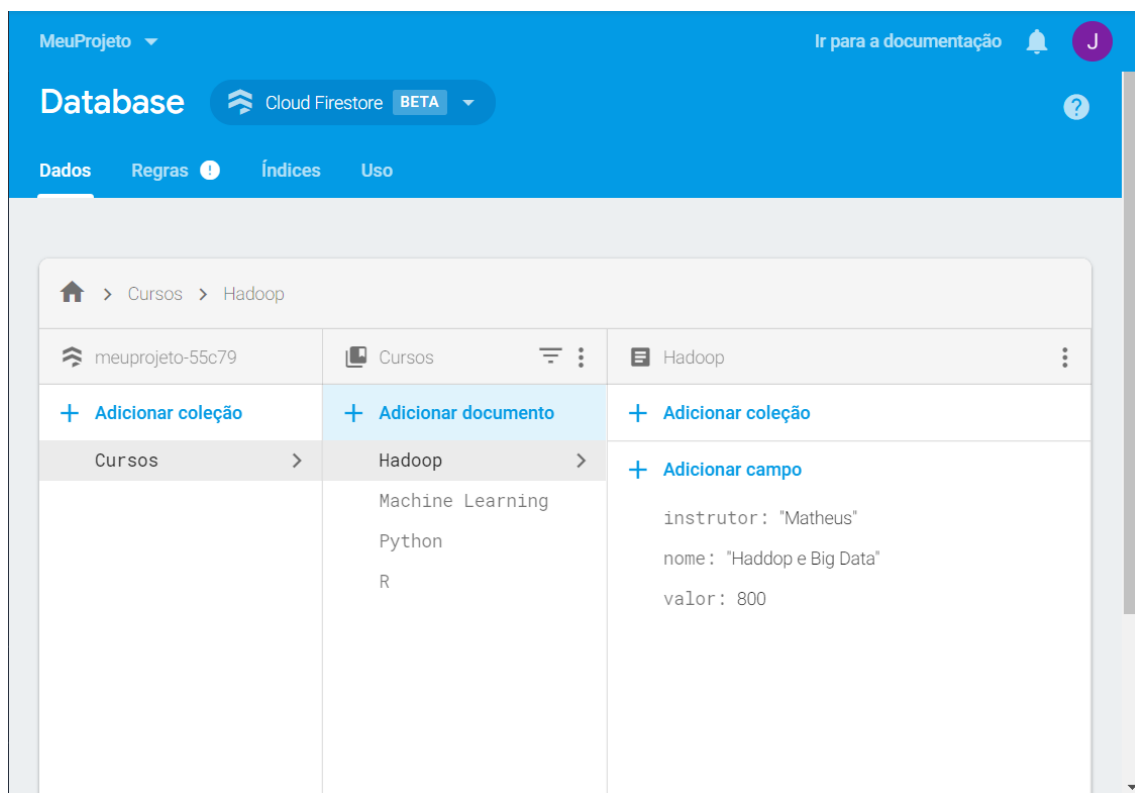
#### Machine Learning

Campos: nome=Machine learning para análise de dados, valor=500 e instrutor= Leo

#### Hadoop

Campos: nome=Hadoop e Big Data, valor=800 e instrutor= Matheus

Veja a tela a seguir



### Instalando o NodeJS e o Firebase -Tools

Antes de instalar o CLI – Command Line Interface na sua máquina, instale o Node.js. Depois disso use o npm para instalar a CLI executando este comando

**npm install -g firebase-tools**

```
PS C:\firebase> firebase init

#####  #####  #####  ###  #####  #####
##  ##  ##  ##  ##  ##  ##  ##
#####  ##  #####  #####  #####  #####  #####
##  ##  ##  ##  ##  ##  ##  ##
##  #####  #####  #####  #####
#####  #####  #####  #####

You're about to initialize a Firebase project in this directory:

  C:\firebase

Before we get started, keep in mind:

  * You are currently outside your home directory
  * You are initializing in an existing Firebase project directory

Are you ready to proceed? Yes
Which Firebase CLI features do you want to setup for this folder? Press Space to select features, then Enter to confirm your choices.
(*) Database: Deploy Firebase Realtime Database Rules
(*) Firestore: Deploy rules and create indexes for Firestore
(*) Functions: Configure and deploy Cloud Functions
( ) Hosting: Configure and deploy Firebase Hosting sites
> (*) Storage: Deploy Cloud Storage security rules
```



```
Administrador: Windows PowerShell

and publish them with firebase deploy.

> What file should be used for Firestore Rules? firestore.rules

Firestore indexes allow you to perform complex queries while
maintaining performance that scales with the size of the result
set. You can keep index definitions in your project directory
and publish them with firebase deploy.

> What file should be used for Firestore indexes? firestore.indexes.json

=== Functions Setup

A functions directory will be created in your project with a Node.js
package pre-configured. Functions can be deployed with firebase deploy.

> What language would you like to use to write Cloud Functions? JavaScript
> Do you want to use ESLint to catch probable bugs and enforce style? No
> File functions/package.json already exists. Overwrite? No
i Skipping write of functions/package.json
> File functions/index.js already exists. Overwrite? No
i Skipping write of functions/index.js
> Do you want to install dependencies with npm now? Yes
npm WARN hoek@2.16.3: The major version is no longer supported. Please update to 4.x or newer

> grpc@1.7.3 install C:\firebase\functions\node_modules\grpc
> node-pre-gyp install --fallback-to-build --library=static_library

[grpc] Success: "C:\firebase\functions\node_modules\grpc\src\node\extension_binary\node-v59-win32-x64-unknown\grpc_node.
node" is installed via remote

> protobufjs@6.8.8 postinstall C:\firebase\functions\node_modules\protobufjs
> node scripts/postinstall

npm notice created a lockfile as package-lock.json. You should commit this file.
added 385 packages in 136.082s

=== Storage Setup

Firebase Storage Security Rules allow you to define how and when to allow
uploads and downloads. You can keep these rules in your project directory
and publish them with firebase deploy.

> What file should be used for Storage Rules? storage.rules

i Writing configuration info to firebase.json...
i Writing project information to .firebaserc...

> Firebase initialization complete!
PS C:\firebase>
```

Para selecionar um recurso basta usar a barra de espaço e em seguida dá um Enter.

Agora ele está perguntando a qual projeto do Firebase você quer vincular esse diretório. Em seguida ele irá fazer várias perguntas, inclusive se você quer substituir os arquivos index.js e packger.js, responda não e, nas demais perguntas dê um Enter para aceitar o padrão.

### Deploy para o Google Cloud Functions via Firebase CLI.

Então, abra o arquivo index.js (Na pasta functions) em seu editor de código favorito e faça as mudanças necessárias.

Abra a pasta onde está o arquivo index.js e execute o comando

**firebase deploy --only funtions**

para fazer o deploy

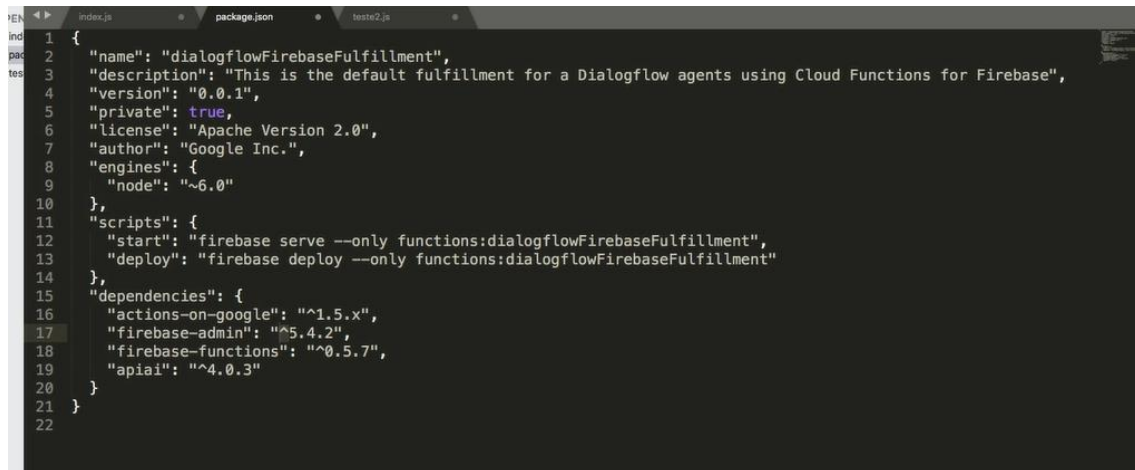
## Integração do Google Cloud Functions para o Cloud Firestore

Abra o arquivo package.json e altere a chave

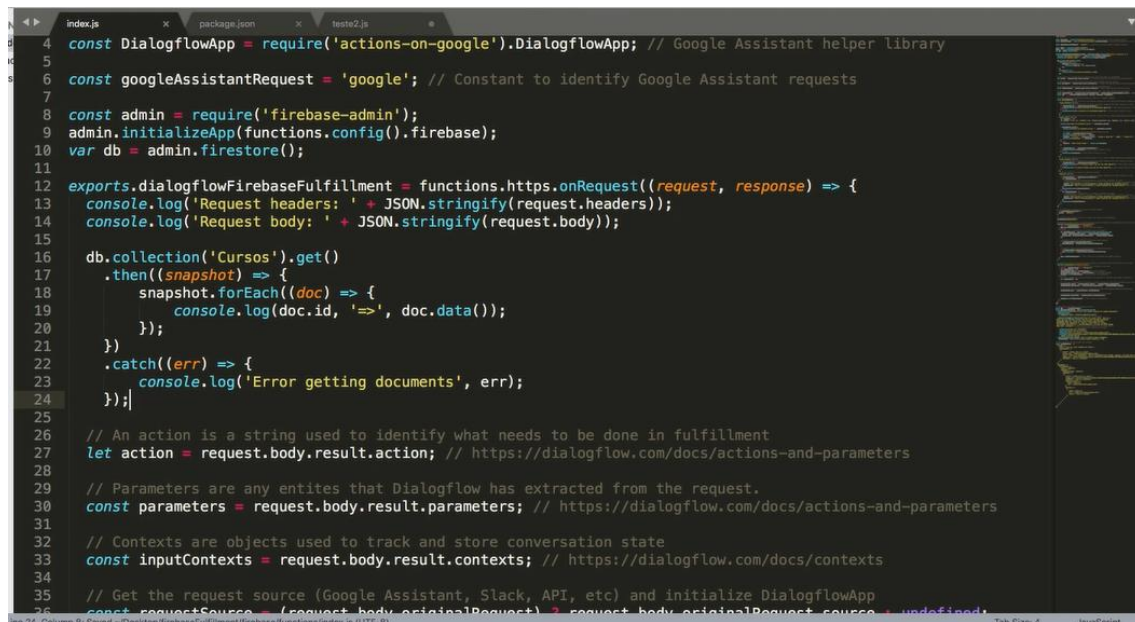
**“firebase-admin”: “^5.4.2”**

Depois vá para o terminal e execute o seguinte comando

**npm install --save firebase-admin**

A screenshot of a code editor showing the package.json file. The file is open in a tab labeled 'package.json'. The code is a JSON object with the following structure: { "name": "dialogflowFirebaseFulfillment", "description": "This is the default fulfillment for a Dialogflow agents using Cloud Functions for Firebase", "version": "0.0.1", "private": true, "license": "Apache Version 2.0", "author": "Google Inc.", "engines": { "node": ">=6.0" }, "scripts": { "start": "firebase serve --only functions:dialogflowFirebaseFulfillment", "deploy": "firebase deploy --only functions:dialogflowFirebaseFulfillment" }, "dependencies": { "actions-on-google": "^1.5.x", "firebase-admin": "^5.4.2", "firebase-functions": "^0.5.7", "apiai": "^4.0.3" } }. The 'firebase-admin' dependency is highlighted in green.

```
1 {
2   "name": "dialogflowFirebaseFulfillment",
3   "description": "This is the default fulfillment for a Dialogflow agents using Cloud Functions for Firebase",
4   "version": "0.0.1",
5   "private": true,
6   "license": "Apache Version 2.0",
7   "author": "Google Inc.",
8   "engines": {
9     "node": ">=6.0"
10  },
11  "scripts": {
12    "start": "firebase serve --only functions:dialogflowFirebaseFulfillment",
13    "deploy": "firebase deploy --only functions:dialogflowFirebaseFulfillment"
14  },
15  "dependencies": {
16    "actions-on-google": "^1.5.x",
17    "firebase-admin": "^5.4.2",
18    "firebase-functions": "^0.5.7",
19    "apiai": "^4.0.3"
20  }
21 }
```

A screenshot of a code editor showing the index.js file. The file is open in a tab labeled 'index.js'. The code is a JavaScript file that uses the actions-on-google library to create a DialogflowApp. It also uses the firebase-admin library to initialize the Firebase app and the firebase-functions library to create a fulfillment function. The code is as follows: 

```
4 const DialogflowApp = require('actions-on-google').DialogflowApp; // Google Assistant helper library
5
6 const googleAssistantRequest = 'google'; // Constant to identify Google Assistant requests
7
8 const admin = require('firebase-admin');
9 admin.initializeApp(functions.config().firebase);
10 var db = admin.firestore();
11
12 exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
13   console.log('Request headers: ' + JSON.stringify(request.headers));
14   console.log('Request body: ' + JSON.stringify(request.body));
15
16   db.collection('Cursos').get()
17     .then((snapshot) => {
18       snapshot.forEach((doc) => {
19         console.log(doc.id, '=>', doc.data());
20       });
21     })
22     .catch((err) => {
23       console.log('Error getting documents', err);
24     });
25
26   // An action is a string used to identify what needs to be done in fulfillment
27   let action = request.body.result.action; // https://dialogflow.com/docs/actions-and-parameters
28
29   // Parameters are any entities that Dialogflow has extracted from the request.
30   const parameters = request.body.result.parameters; // https://dialogflow.com/docs/actions-and-parameters
31
32   // Contexts are objects used to track and store conversation state
33   const inputContexts = request.body.result.contexts; // https://dialogflow.com/docs/context
34
35   // Get the request source (Google Assistant, Slack, API, etc) and initialize DialogflowApp
36   const requestSource = (request.body.originalRequest ? request.body.originalRequest.source : undefined);
```

Faça o deploy

**firebase deploy --only functions**

```

4 const DialogflowApp = require('actions-on-google').DialogflowApp; // Google Assistant helper library
5
6 const googleAssistantRequest = 'google'; // Constant to identify Google Assistant requests
7
8 const admin = require('firebase-admin');
9 admin.initializeApp(functions.config().firebase);
10 var db = admin.firestore();
11
12 exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
13   console.log('Request headers: ' + JSON.stringify(request.headers));
14   console.log('Request body: ' + JSON.stringify(request.body));
15
16   /*db.collection('Cursos').get()
17   .then((snapshot) => {
18     snapshot.forEach((doc) => {
19       console.log(doc.id, '=>', doc.data());
20     });
21   })
22   .catch((err) => {
23     console.log('Error getting documents', err);
24   });*/
25
26   // An action is a string used to identify what needs to be done in fulfillment
27   let action = request.body.result.action; // https://dialogflow.com/docs/actions-and-parameters
28
29   // Parameters are any entities that Dialogflow has extracted from the request.
30   const parameters = request.body.result.parameters; // https://dialogflow.com/docs/actions-and-parameters
31
32   // Contexts are objects used to track and store conversation state
33   const inputContexts = request.body.result.contexts; // https://dialogflow.com/docs/context
34
35   // Get the request source (Google Assistant, Slack, API, etc) and initialize DialogflowApp
36   const requestSource = (request.body.originalRequest ? request.body.originalRequest.source : undefined);

```

```

50 // The default welcome intent has been matched, welcome the user (https://dialogflow.com/docs/events#default-welcome)
51 'preco.curso': () => {
52   var resposta = "";
53   var cursos = {'R': 100, 'Python': 200, 'Machine Learning': 300, 'Hadoop': 400, 'Scala': 500};
54
55   console.log("valor do parametro curso: " + parameters.curso);
56
57   if(parameters.curso) {
58     /*console.log("Valor do parametro curso: " + parameters.curso);
59
60     // Parte 1 - trazendo informações buscando dados do array
61     var valor = cursos[parameters.curso];
62     console.log("Valor do curso: " + valor);
63     resposta = 'O curso ' + parameters.curso + " possui o valor de " + valor + " reais!!!!";
64     console.log("Resposta: " + resposta);*/
65
66     var docRef = db.collection("Cursos").doc(parameters.curso);
67
68     docRef.get().then(function(doc) {
69       console.log(doc.data());
70       resposta = 'O curso ' + parameters.curso + " possui o valor de " + doc.data().Valor + " reais.";
71       if (requestSource === googleAssistantRequest) {
72         sendGoogleResponse(resposta); // Send simple response to user
73       } else {
74         sendResponse(resposta); // Send simple response to user
75       }
76     }).catch(function(error) {
77       console.log("Error getting document:", error);
78     });
79
80   }
81   else {
82     resposta = "Qual curso? Temos: " + Object.keys(cursos);

```

```

79
80   }
81   else {
82     resposta = "Qual curso? Temos: " + Object.keys(cursos);
83   }
84
85   if (requestSource === googleAssistantRequest) {
86     sendGoogleResponse(resposta); // Send simple response to user
87   } else {
88     sendResponse(resposta); // Send simple response to user
89   }
90 },
91 // The default fallback intent has been matched, try to recover (https://dialogflow.com/docs/i

```

## Código com as alterações até o momento

```
1 // See https://github.com/dialogflow/dialogflow-fulfillment-nodejs
2 // for Dialogflow fulfillment library docs, samples, and to report issues
3 'use strict';
4
5 const functions = require('firebase-functions');
6 const {WebhookClient} = require('dialogflow-fulfillment');
7 const {Card, Suggestion} = require('dialogflow-fulfillment');
8 const DialogflowApp = require('actions-on-google').DialogflowApp;
9
10 const googleAssistantRequest = 'google';
11
12 const admin = require('firebase-admin');
13 admin.initializeApp(functions.config().firebase);
14 var db = admin.firestore();
15
16 process.env.DEBUG = 'dialogflow:debug'; // enables lib debugging statements
17
18 exports.dialogflowFirebaseFulfillment = functions.https.onRequest((request, response) => {
19   const agent = new WebhookClient({ request, response });
20   console.log('Dialogflow Request headers: ' + JSON.stringify(request.headers));
21   console.log('Dialogflow Request body: ' + JSON.stringify(request.body));
22
23   db.collection('Cursos').get()
24     .then((snapshot) => {
25       snapshot.forEach((doc) => {
26         console.log(doc.id, '=>', doc.data());
27       });
28     })
29     .catch((err) => {
30       console.log('Error getting documents', err);
31     });
32
33   let action = request.body.result.action;
34
35   const parameters = request.body.result.parameters;
36
37   const inputContexts = request.body.result.contexts;
38
39   const requestSource = (request.body.originalRequest) ? request.body.originalRequest.source : undefined;
40   const app = new DialogflowApp ({request: request, response: response});
41
42   const actionHandlers = {
43     'input.welcome': () => {
44       if (requestSource === googleAssistantRequest){
45         sendGoogleResponse('Olá, Bem-vindo ao assistente de consulta.');
```

```

66     }).catch(function(err){
67         console.log('Error getting document', err);
68     });
69     } else {
70         resposta = "Qual curso? Temos: " + object.keys(cursos);
71         if (requestSource === googleAssistantRequest) {
72             sendGoogleResponse(resposta);
73         } else {
74             sendGoogleResponse(resposta);
75         }
76     }
77 },
78 }
79 function welcome(agent) {
80     agent.add('Welcome to my agent!');
81 }
82
83 function fallback(agent) {
84     agent.add('I didn't understand');
85     agent.add('I'm sorry, can you try again?');
86 }
87
88 function sendGoogleResponse(responseToUser) {
89     if (typeof responseToUser === 'string'){
90         app.ask(responseToUser);
91     } else {
92     }
93 }
94 }

```

```

95 // // Uncomment and edit to make your own intent handler
96 // // uncomment `intentMap.set('your intent name here', yourFunctionHandler);`
97 // // below to get this function to be run when a Dialogflow intent is matched
98 // function yourFunctionHandler(agent) {
99 //     agent.add('This message is from Dialogflow's Cloud Functions for Firebase editor!');
100 //     agent.add(new Card({
101 //         title: `Title: this is a card title`,
102 //         imageUrl: 'https://developers.google.com/actions/images/badges/XPM_BADGING_GoogleAssistant_VER.png',
103 //         text: `This is the body text of a card. You can even use line\n breaks and emoji! 🍌`,
104 //         buttonText: 'This is a button',
105 //         buttonUrl: 'https://assistant.google.com/'
106 //     }));
107 // });
108 // agent.add(new Suggestion('Quick Reply'));
109 // agent.add(new Suggestion('Suggestion'));
110 // agent.setContext({ name: 'weather', lifespan: 2, parameters: { city: 'Rome' }});
111 // }
112
113 // // Uncomment and edit to make your own Google Assistant intent handler
114 // // uncomment `intentMap.set('your intent name here', googleAssistantHandler);`
115 // // below to get this function to be run when a Dialogflow intent is matched
116 // function googleAssistantHandler(agent) {
117 //     let conv = agent.conv(); // Get Actions on Google library conv instance
118 //     conv.ask('Hello from the Actions on Google client library!') // Use Actions on Google library
119 //     agent.add(conv); // Add Actions on Google library responses to your agent's response
120 // }
121 // // See https://github.com/dialogflow/dialogflow-fulfillment-nodejs/tree/master/samples/actions-on-google
122 // // for a complete Dialogflow fulfillment library Actions on Google client library v2 integration sample
123
124 // Run the proper function handler based on the matched Dialogflow intent name
125 let intentMap = new Map();
126 intentMap.set('Default Welcome Intent', welcome);

```

```

127 intentMap.set('Default Fallback Intent', fallback);
128 // intentMap.set('your intent name here', yourFunctionHandler);
129 // intentMap.set('your intent name here', googleAssistantHandler);
130 agent.handleRequest(intentMap);
131 });
132

```

```

65
66     var docRef = db.collection("Cursos").doc(parameters.curso);
67
68     docRef.get().then(function(doc) {
69         console.log(doc.data());
70         resposta = '0 curso ' + parameters.curso + ' possui o valor de ' + doc.data().Valor + ' reais.';
71         if (requestSource === googleAssistantRequest) {
72             sendGoogleResponse(resposta); // Send simple response to user
73         } else {
74             sendResponse(resposta); // Send simple response to user
75         }
76     }).catch(function(error) {
77         console.log("Error getting document:", error);
78     });
79
80 }
81 else {
82     resposta = "Qual curso? Temos: " + Object.keys(cursos);
83     if (requestSource === googleAssistantRequest) {
84         sendGoogleResponse(resposta); // Send simple response to user
85     } else {
86         sendResponse(resposta); // Send simple response to user
87     }
88 }
89

```