**Plano de aula 07 – WebHook (Python)**

Nesta aula falaremos sobre WebHooks com Python.

**Passo 1**

**Criando uma conta no Heroku**

Para criar uma conta no Heroku acesse o site (<https://www.heroku.com>). Não esqueça de selecionar a linguagem primária como sendo Python.

**Passo 2**

**Criando uma aplicação no Heroku via GitHub**

Vamos criar uma aplicação no Heroku a partir de um código fonte já disponível no GitHub. Para isto, vá até o link <https://github.com/svet4/shipping-costs-sample> .

Na seção inferior da página tem um botão **Deploy no Heroku**. Então clique nele para fazer o deploy.

Especifique o nome do aplicativo, por exemplo **ifrn-bot-teste**, e clique no botão **Deploy**.

**Clonando o repositório do GitHub.**

Acesse seu aplicativo (O que você acabou de criar) no dashboard do aplicativo <https://dashboard.heroku.com/apps/ifrn-bot-teste/deploy/heroku-git> , clique na opção **Deploy** e instale o Heroku CLI. Através da opção **install Heroku CLI.**

Para fazer a instalação do Heroku CLI, abra um terminal e execute o comando:

**install heroku**

após instalação, conecte-se ao heroku

**$ heroku login**

**Clonando o repositório do GitHub**

Crie uma pasta para o bot e, nessa pasta execute o comando

**mkdir Bots**

**cd Bots**

**$ heroku git:clone -a nome-de-seu-bot**

Não se esqueça que o “**nome-de-seu-bot”** é o nome da aplicação que você fez o deploy no heroku.

Você clonou o repositório, porém ele está vazio.

Agora vamos baixar o código fonte para esse diretório, portanto, execute o comando

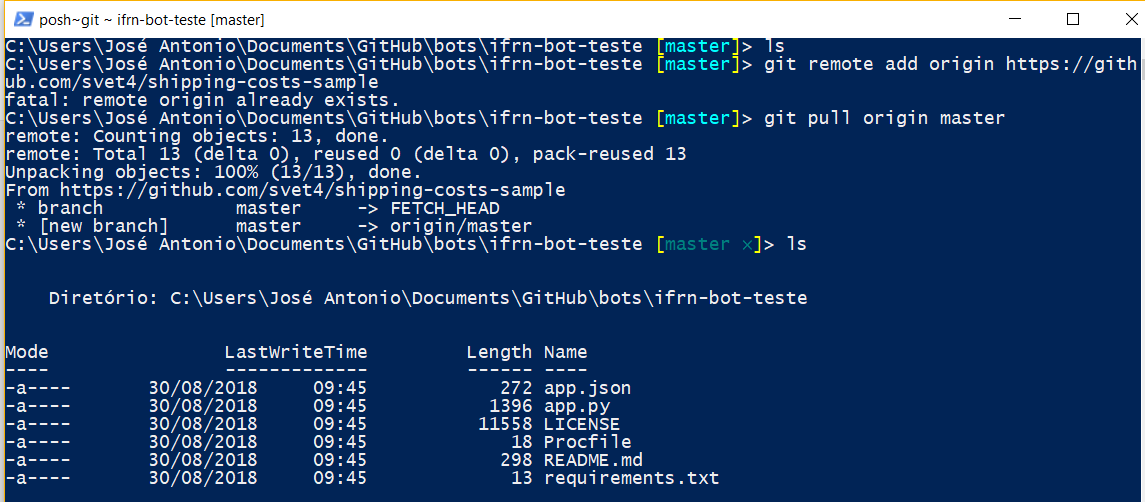
**cd ifrn-bot-teste** //para entrar para a pasta

**git remote add origin** [**https://github.com/svet4/shipping-costs-sample**](https://github.com/svet4/shipping-costs-sample)

Agora execute o comando a seguir para baixar os arquivos.

**git pull origin master**

Agora temos 6 arquivos, como mostra a Figura 1.



**Figura 1. Arquivos baixados**.

**Entendo o código app.py**

Abra o arquivo app.py e dê uma olhada no código para entendê-lo. Observe que o mesmo trata de preço de um frete de uma determinada região. Como mostra a Figura 2.



Figura 2. Arquivo app.py

**Integrando o Dialogflow como o Heroku**

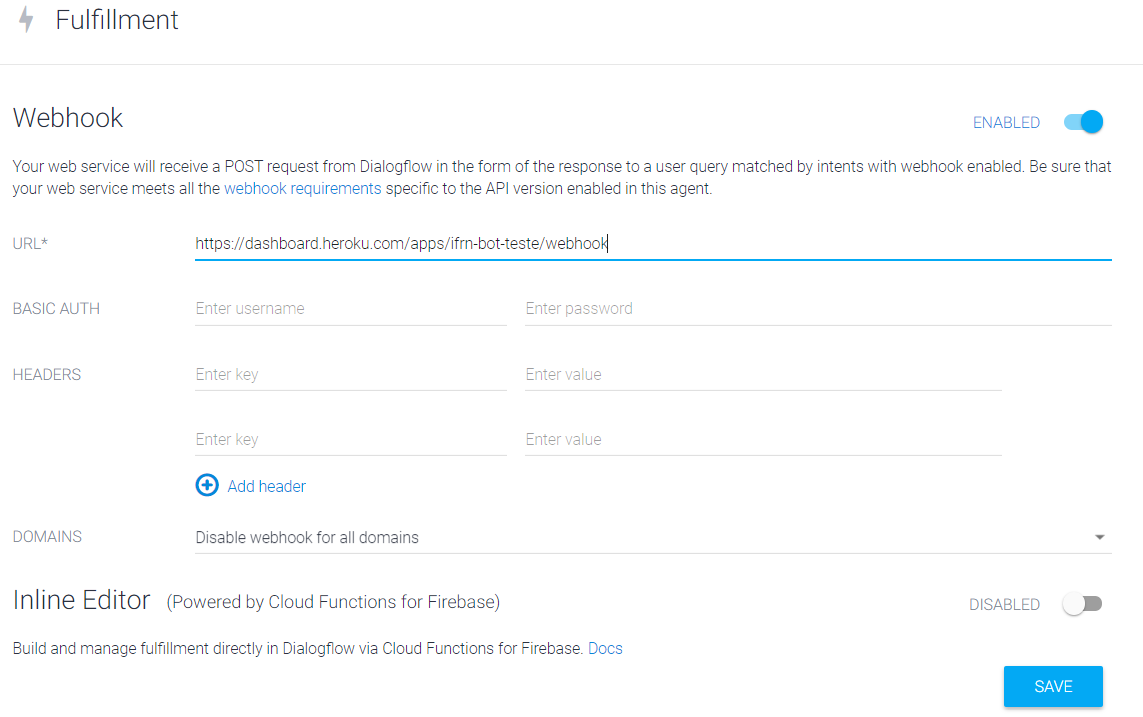
No código anterior, podemos ver que está sendo uma verificação se a “action”, é “shipping.cost”. Então, precisamos alterar essa ação para “**preco.curso**”.

**Criando uma Intenção para utilização de WebHook**

Inicialmente crie uma entidade **curso** para representar os cursos oferecidos.

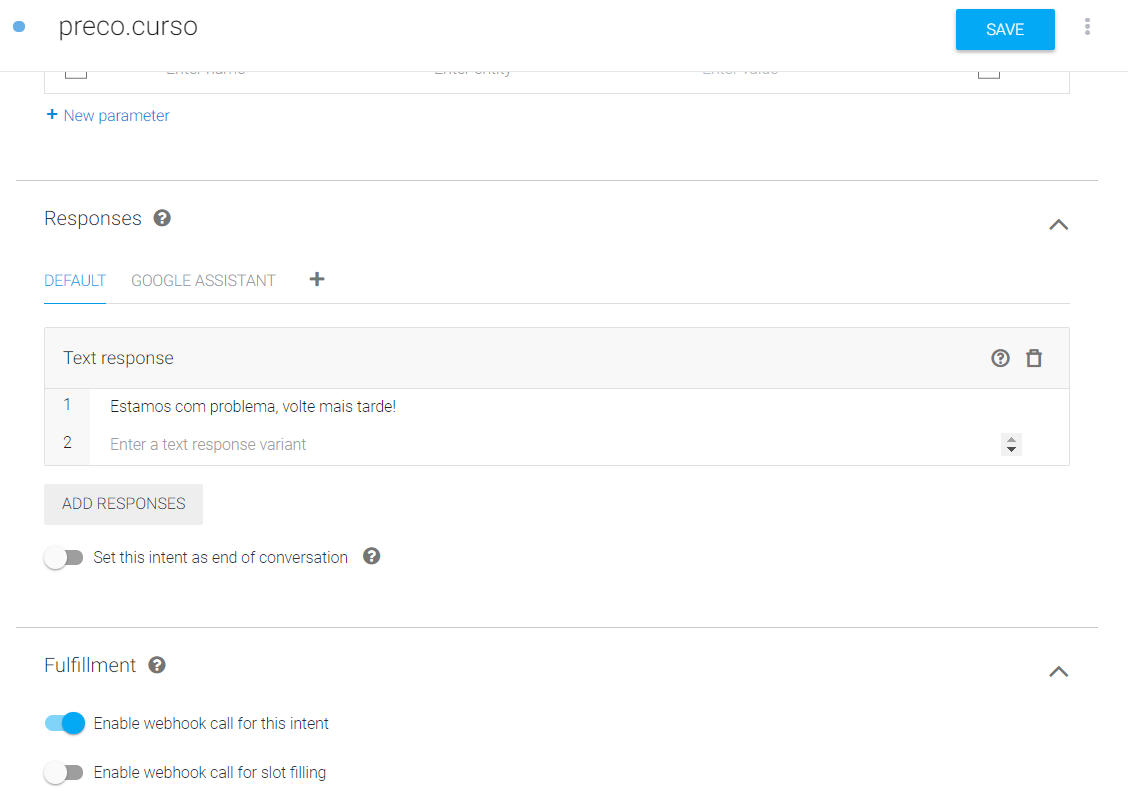
* Python, Machine Learning 🡪 ML, R e Hadoop
* Crie uma intente chamada **curso.valor**, e entre com as seguintes frases para treinamento:
  + Qual o preço do curso de Machine Learning?
  + Quanto custa o curso de R?
  + Qual o valor do curso de Python?
  + @ Gostaria de saber o preço do curso @curso.curso
* **Resposta padrão**: O valor do curso $curso é de 100 reais.
* Antes de integrar ao **backend**, salve e faças alguns testes.

Precisamos ir até a aba de fulfilment e habilitar o WebHook e informar a URL de nossa aplicação, como mostra a Figura 3.



**Figura 3. Habilitando WebHook**

Ainda precisamos habilitar o webhook para a intente que queremos utilizá-lo. Nem toda intente usa webhook. Veja a Figura 4.



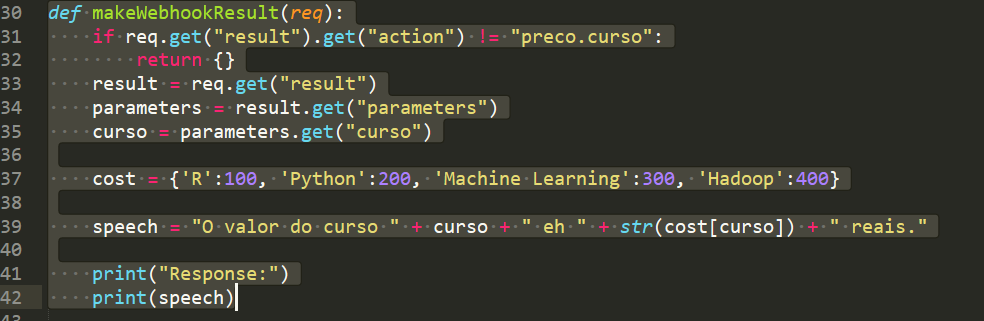
**Figura 4. Habilitando a intenção para usar WebHook.**

Salve e faça um teste. Observe que no json a seguir temos a **action = preco.curso**



**Alterando o Código Fonte do BackEnd**

Vamos fazer as seguintes alterações no código fonte, como mostra a Figura 5.



**Figura 5. Alterando o código fonte para trabalhar com preço de cursos.**

**Testando as alterações no código fonte antes de fazer o deploy**

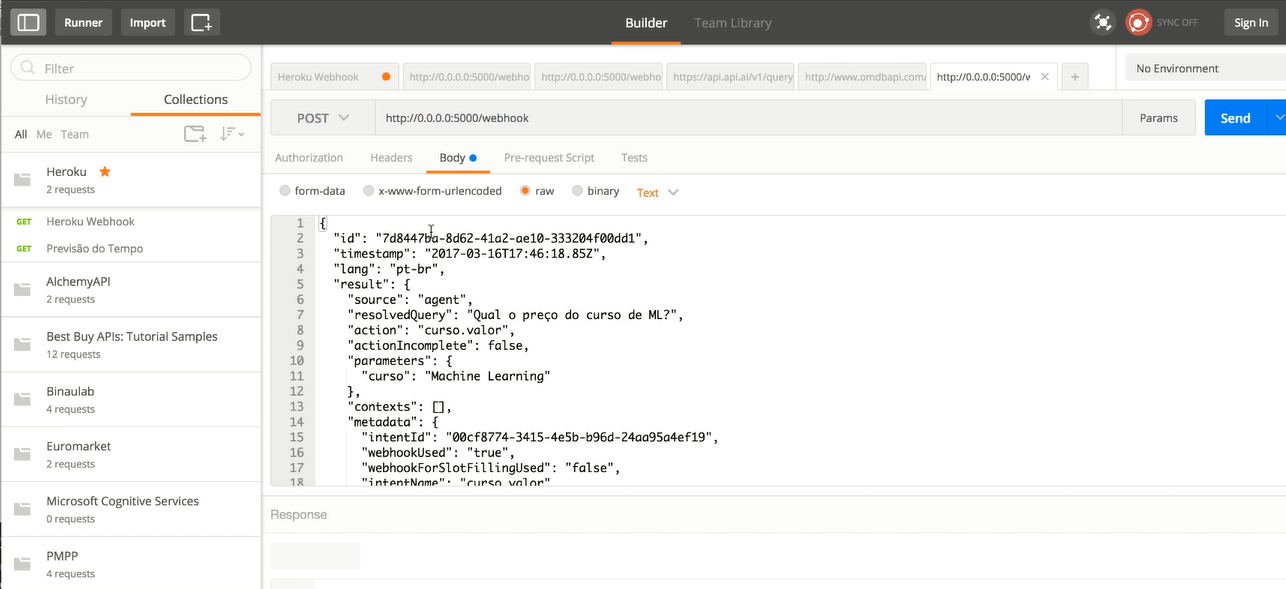
Se você estiver usando o Sublime text. Você pode compilar a aplicação e testar no Postman, por exemplo. Veja a Figura 6.



**Figura 6. Compilando a aplicação app.py no Sublime Text**.

Na opção Tools selecione Build. Se não tiver erro, o sistema exibe que a aplicação está rodando em <http://0.0.0.0:5000>

A partir desse momento, podemos disparar uma requisição POST. Vamos fazer isso usando o **Postman**. No Postman, selecione o verbo Post e informe a URL <http://127.0.0.1:5000/webhook> e no Body cole o json copiado do DialogFlow. Veja a Figura 7.



**Figura 7**. Rodando a aplicação no Postman.

**Fazendo o Deploy de uma nova versão**

Fazendo o deploy para o servidor.

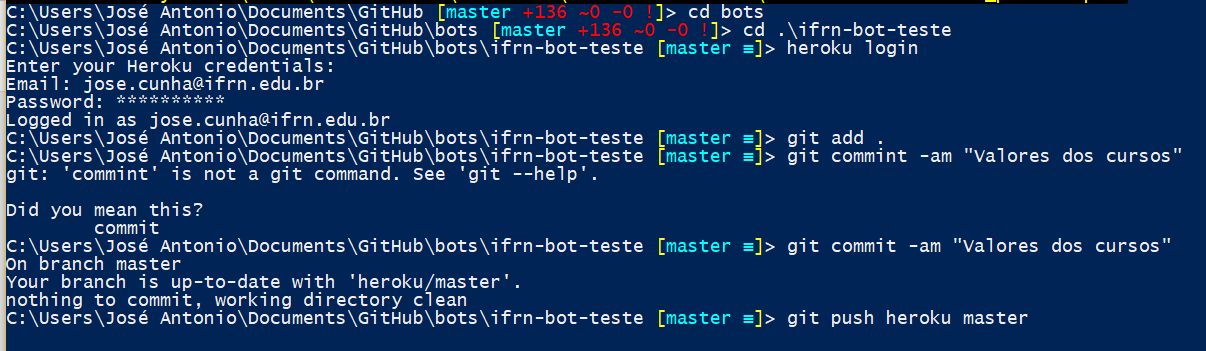
**cd bots**

**cd ifrn-bot-teste**

**git add .**

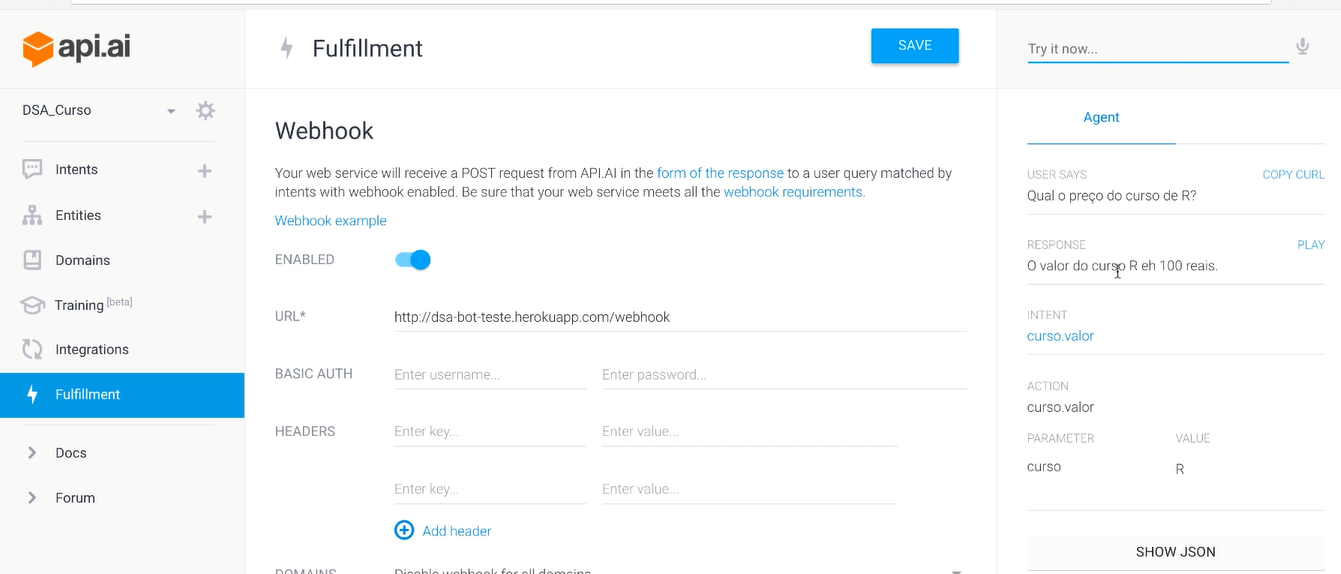
**git commit -am “Valores dos cursos”**

**git push heroku master**



**Figura 8. Fazendo o deploy da aplicação app.py via terminal.**

Se tudo correr bem! Você agora pode testar seu bot diretamente da ferramenta DialogFlow.



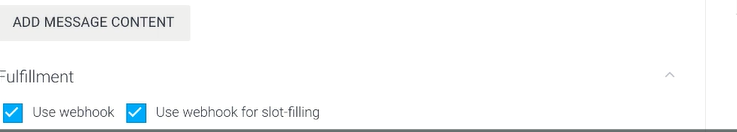
**Figura 9. Testando o bot no DialogFlow**.

**WebHook for Slot Filling**

Suponha que o usuário não informe o curso. Por exemplo, se o mesmo fizer a seguinte pergunta “qual o valor do curso?”

Uma primeira tentativa seria tornar o parâmetro curso como obrigatório e definir um prompt “Qual curso? Escolha entre R, Python, Machine Learning e Hadoop:”

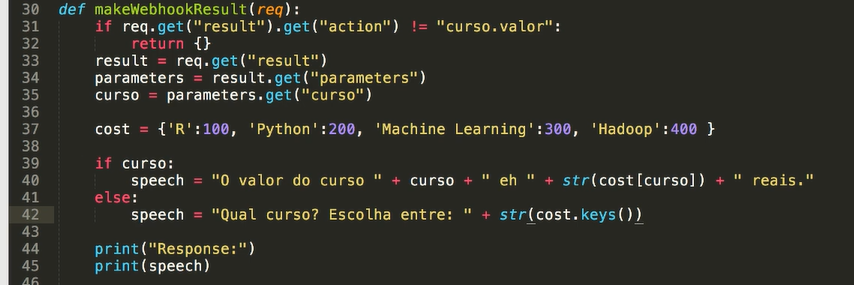
Agora, marque a opção “use Webhook slot filling”, como mastra a Figura 10.



**Figura 10**. Habilitando WebHook for slot-filling

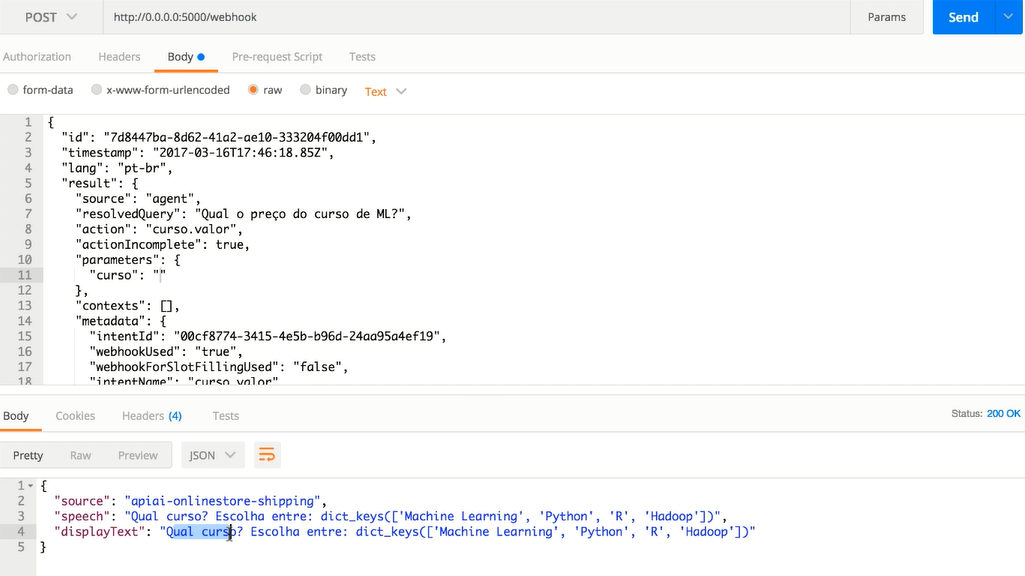
Habilitando **WebHooks Slot Filling** no **BackEnd**

Uma opção é analisar se o parâmetro **curso** está vazio e retornar uma informação para o bot.



**Figura 11. Usando WebHook Slot Filling no BackEnd.**

Salve e teste. Podemos testar antes de fazer o deploy. Sim, vá até o Postman e mude o valor da propriedade actionIncomplete para true, como mostra a Figura 12.



**Figura 12. Rodando no Postman**.

Veja que o prompt retornado foi do BackEnd.

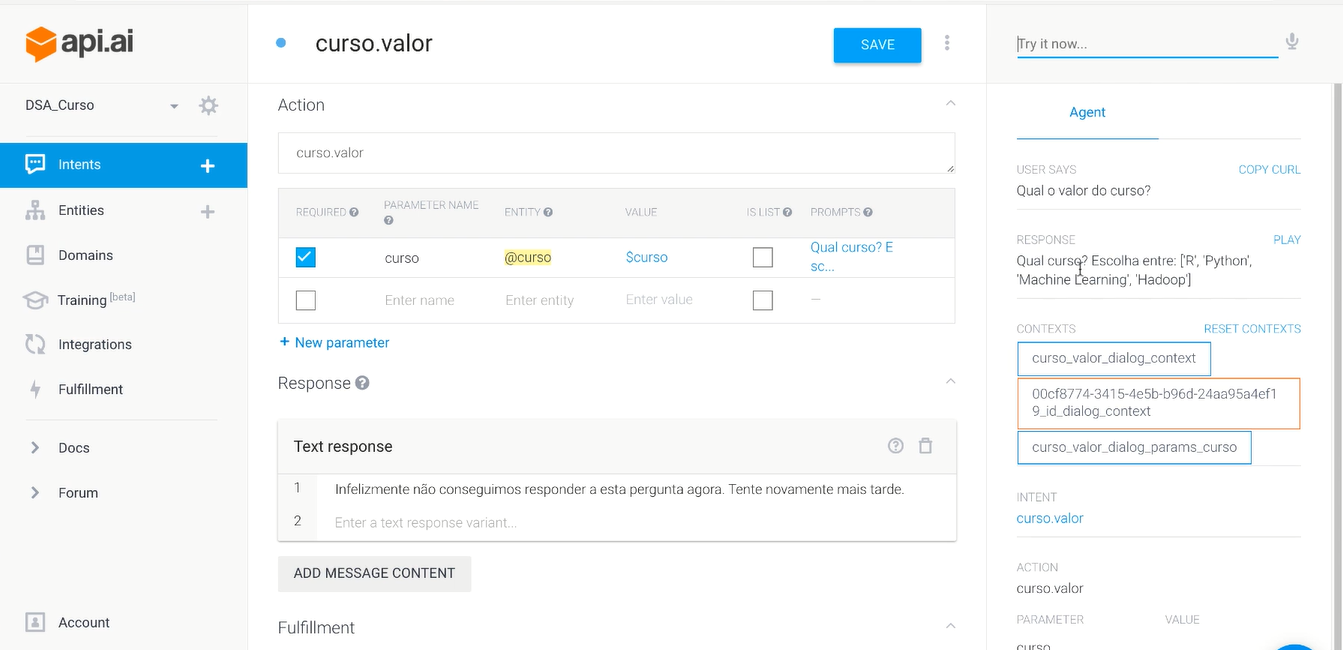
Fazendo o deploy para atualizar o código no servidor.

**git add .**

**git commit -am “Curso valor slot filling”**

**git push heroku master**

**Testando no DialogFlow**

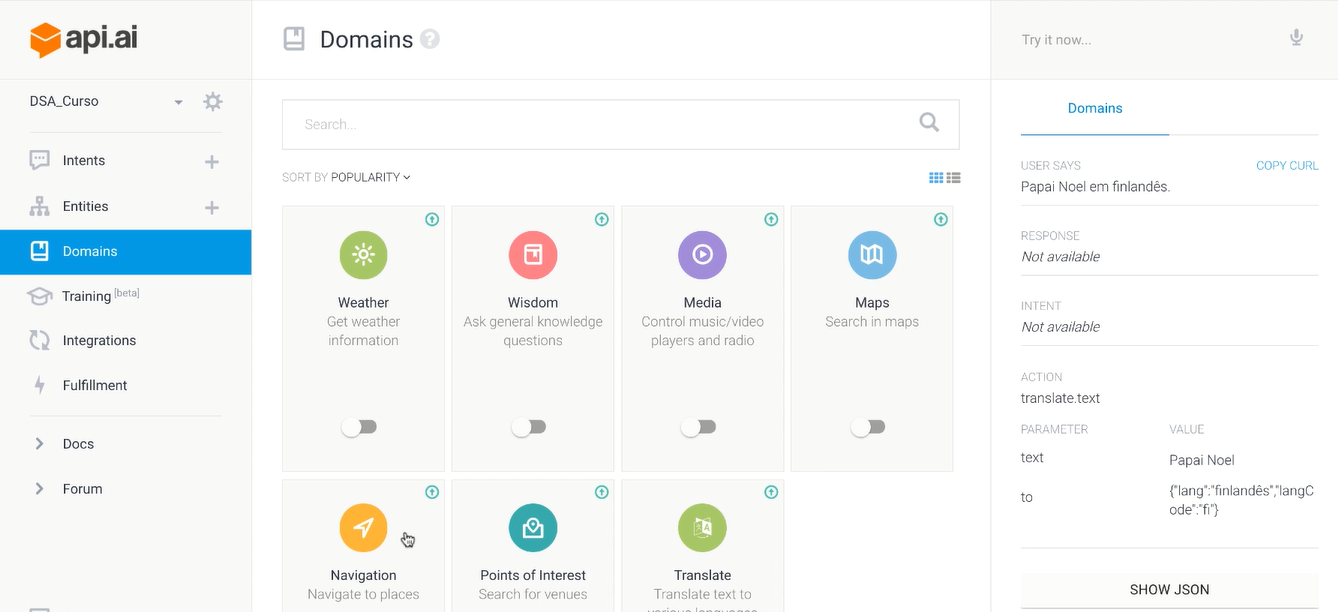


**Figura 13**. Testando as alterações no DialogFlow.

Observe que o prompt exibido para o usuário é do servidor.

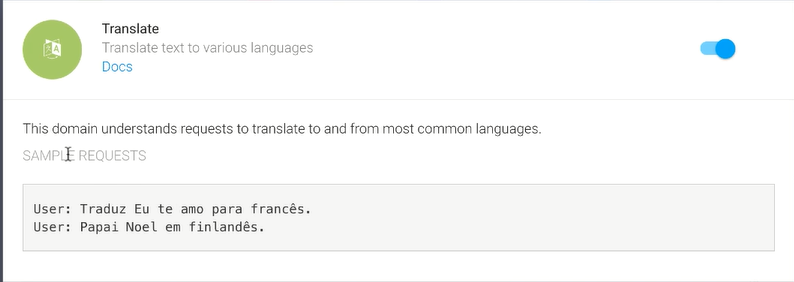
**WebHook for Domains**

Mais o que são domínios. Domínios são pacotes de conhecimentos já pré-definidos por áreas onde nós podemos ativar. Por exemplo, se você quer construir um bot que vai auxiliar o usuário em uma navegação, você não precisa fazer todo o caminho da construção de seu bot. Nesse caso, você pode fazer tudo apenas habilitando o Domínio Navegation, como mostra a Figura 14.



**Figura 14.** Biblioteca de Domínios do DialogFlow

Para testar, vamos habilitar o Domínio **Translate**. Clique em **VIEW DETAILS**, será mostrado a Figura 15, onde são exibidas algumas expressões.



**Figura 15**.

Feche essa janela, e faça um teste informando ao bot a frase “Papai Noel em finlandês”.

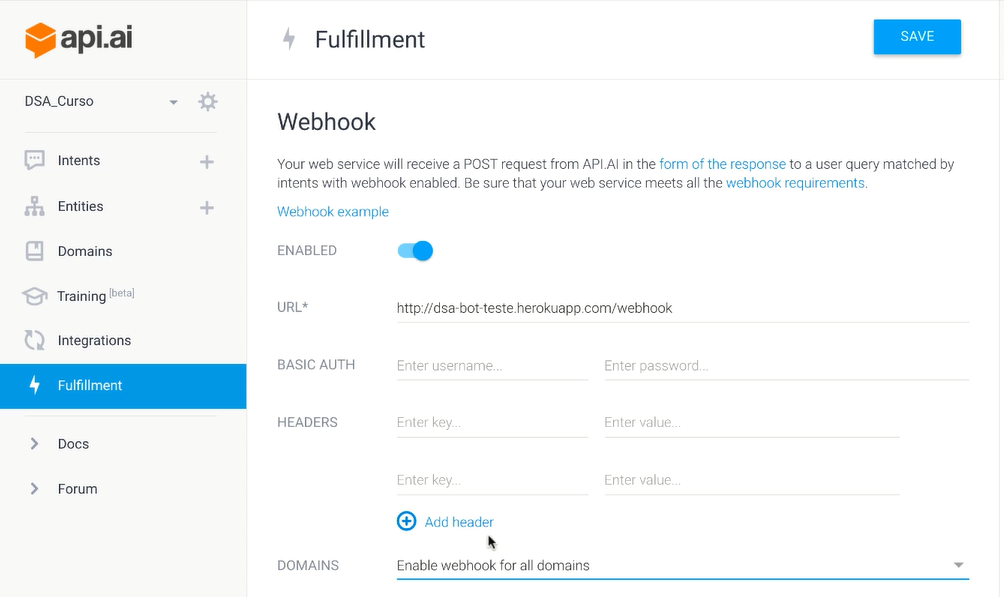
O resultado deve ser como o da Figura 16. Observe as propriedades **text**  e **to**.



**Figura 16. Saída apresentada pelo DialogFlow**.

Agora vamos passar essas informações para o BackEnd, como fazemos?

Vá até Fulfilment e habilite o WebHook para todos os domínios, como mostra a Figura 17.

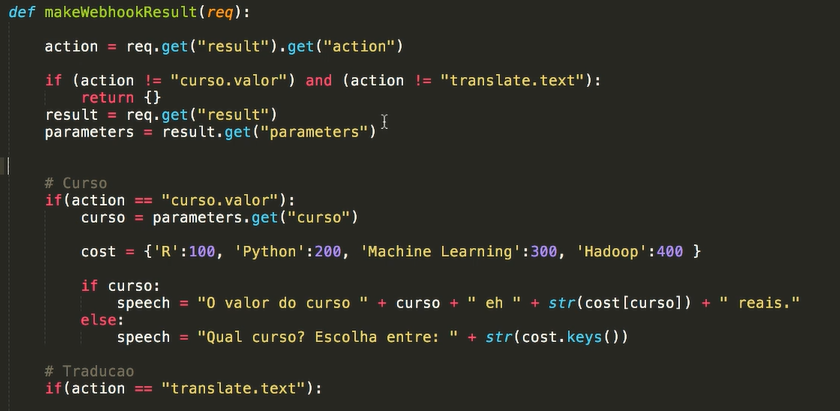


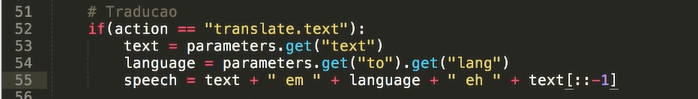
**Figura 17. Habilitando WebHook para todos os Domínios**

Salve.

Só que agora nosso BackEnd tem que processar uma ação chamada “**translate.text**”, que espera dois parâmetros “text” e “to”.

Alterando o BackEnd para utilizar Domínios. Veja a Figura 18.

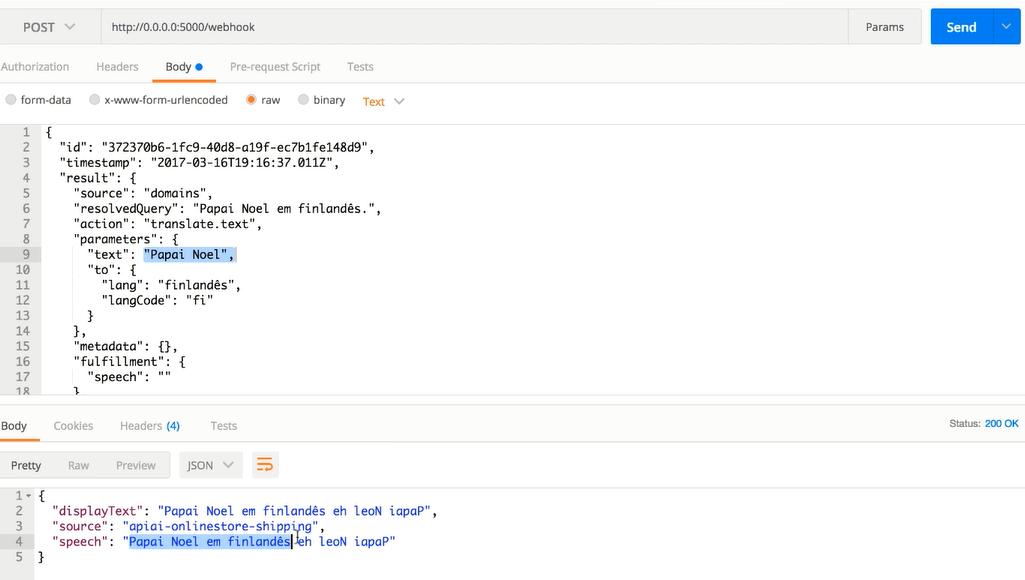




**Figura 18. Código alterado para trabalhar com Domínios.**

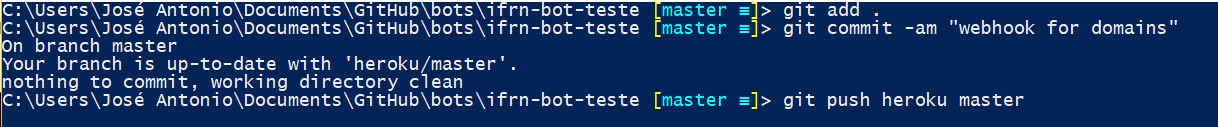
Aqui estou apenas imprimindo o texto na ordem inversa. Mas claro, que você poderia usar uma **API**, por exemplo, do Google Tradutor e fazer a tradução correta.

Salve e teste via Postman antes de fazer o deploy para o servidor.



**Figura 19. Testando a aplicação via Postman**.

Faça o deploy e teste no DialogFlow.



**Firebase**

Vamos voltar ao problema de valores de um curso. Nós temos um problema, toda vez que o valor de um curso alterar, nós temos que alterar nosso código fonte e fazer o deploy, e isso, não é bom. Mais você devia ter imaginado que nós poderíamos utilizar um banco de dados para armazenar esses valores. Pode ser um banco de dados relacional ou não. Para o nosso estudo de caso de exemplo, vamos utilizar o Firebase.

No Firebase temos diversas formas de armazenamento de dados, porém nós utilizaremos para esse exemplo o **RealTime Database**, como mostra a Figura 20.

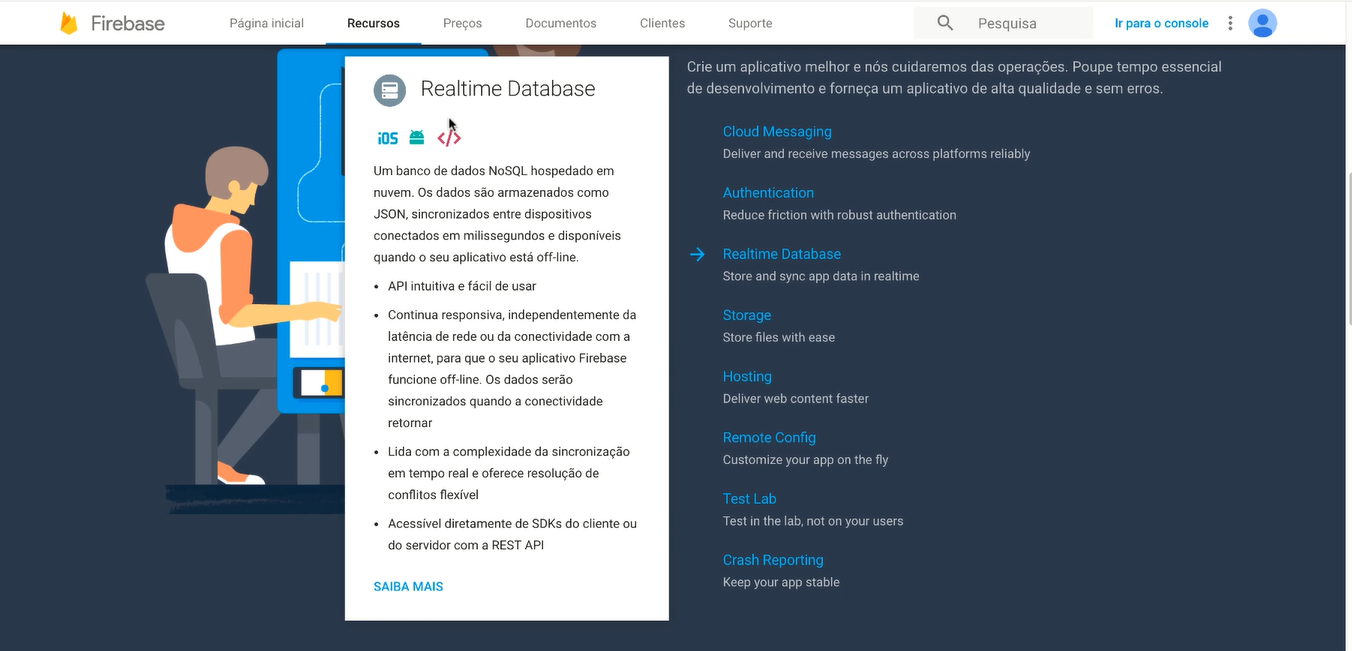
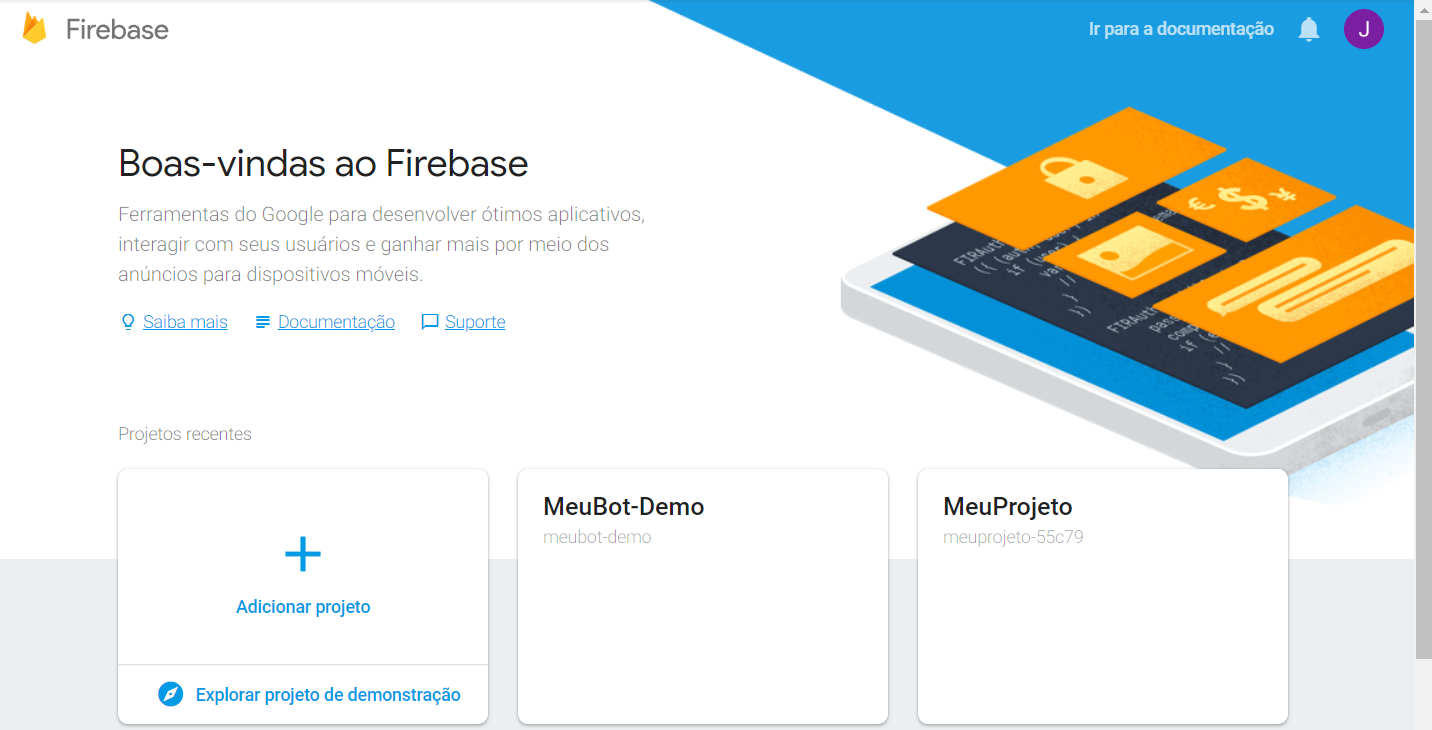


Figura 20.

Então, vamos criar um projeto no Firebase. Na Figura 21, clique no link Adicionar projeto.



**Figura 21. Criando novo projeto no Firebase**.

Na Figura 22, informe o nome do projeto e clique no botão Criar projeto.

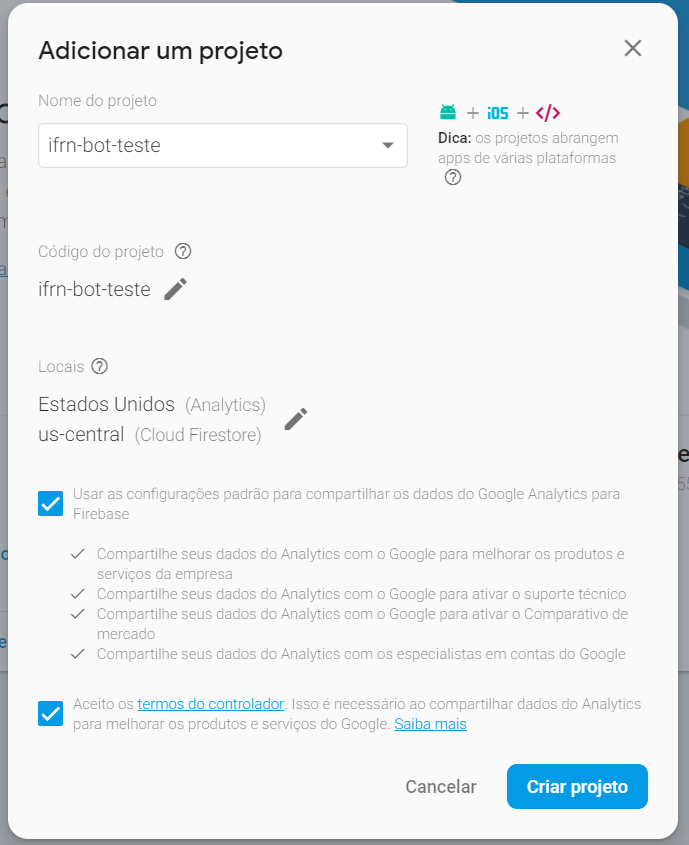


Figura 22.

Após criar o projeto, clique na opção do menu Database, como mostra a figura 23.

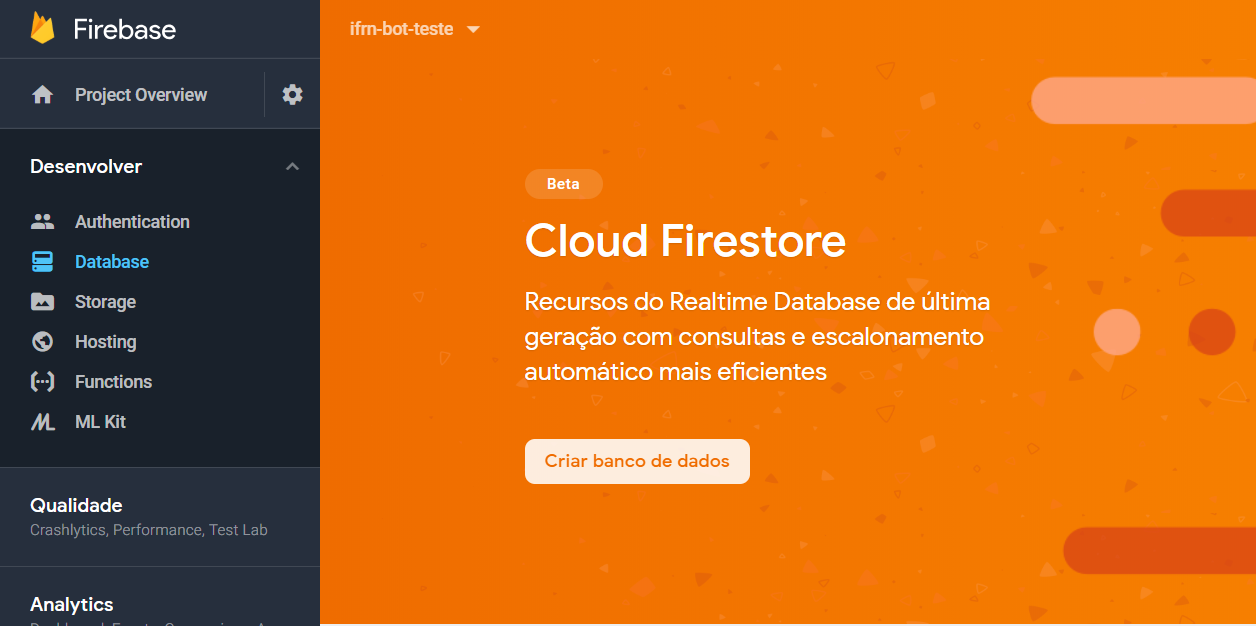


Figura 23.

E clique no botão criar banco de dados e na Figura 24 clique em Ativar.



**Figura 24.**

A figura 25, mostra a tela onde adicionamos novas coleções ao banco de dados.



**Figura 25**. Adicionando coleções ao banco de dados.

**Permissões em Banco de Dados Firebase**

Vamos manipular as permissões ou regras no banco de dados Fierbase. A regra mais geral de todas é mostrada na Figura 26.

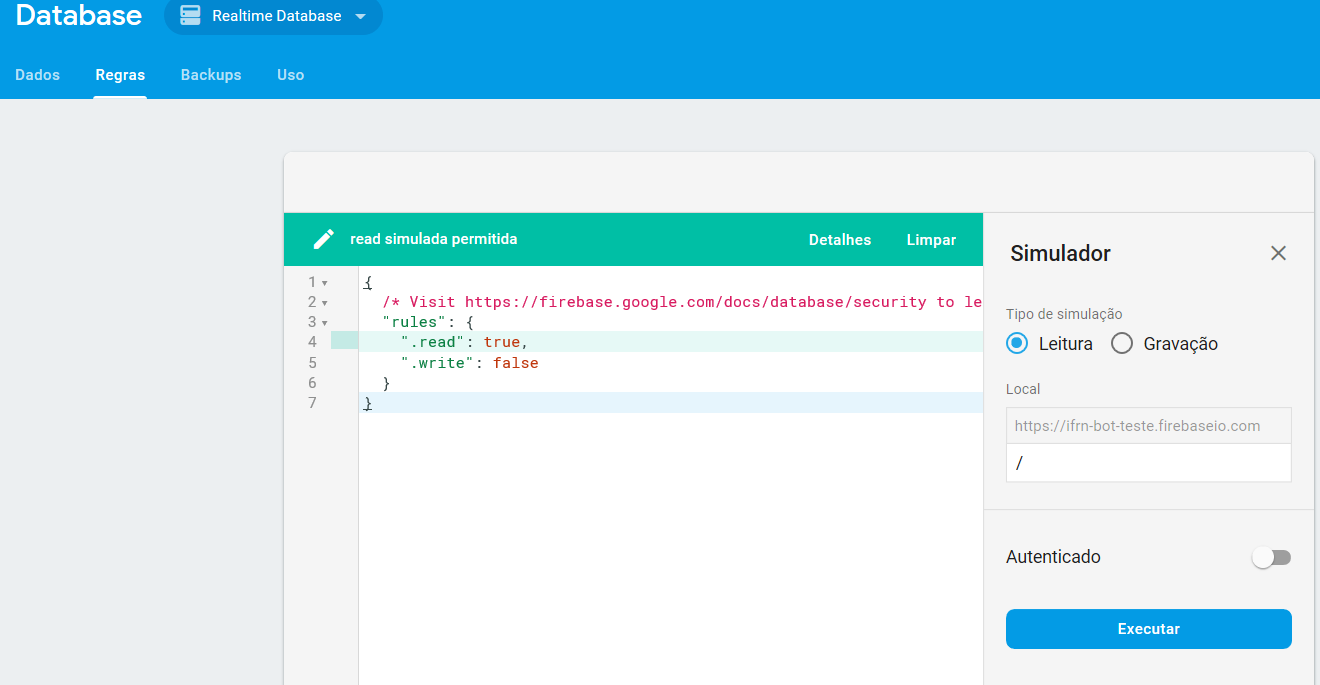


Figura 26. Regras de permissões.

Quando você muda uma regra de permissão, você deve publicá-la. Veja a Figura 27.

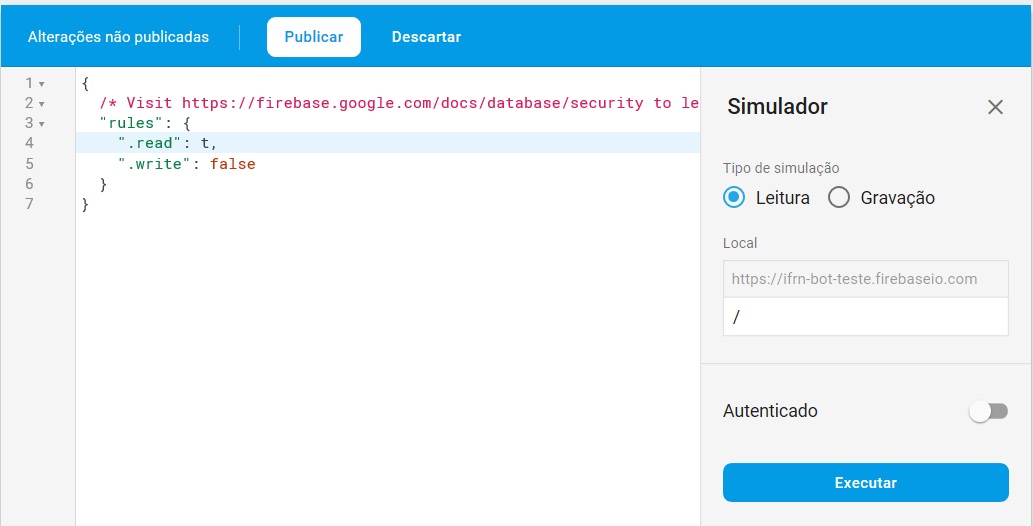


Figura 27. Publicar uma regra de permissão alterada.

Publique as modificações e simule ainda sem autenticar o usuário. Veja a Figura 28.

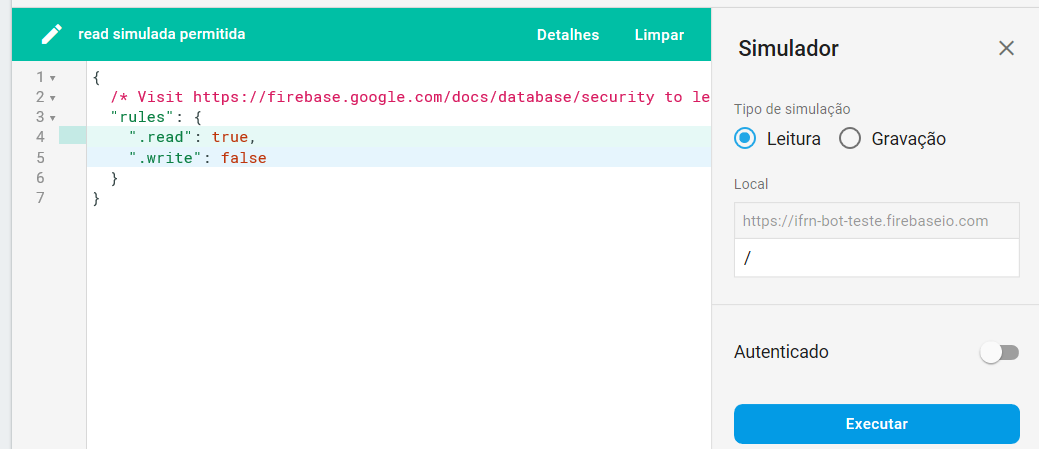


Figura 28.

Como podemos testar isso? Podemos fazer o teste em uma janela anônima. Pegue o endereço URL do banco de dados Firebase em Configurações, como mostra a Figura 29.

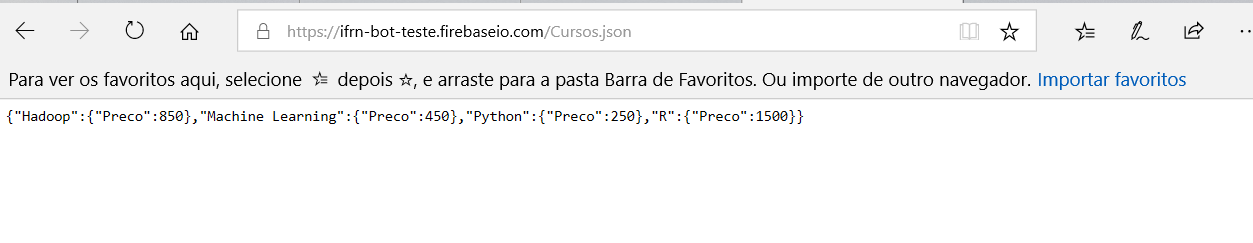
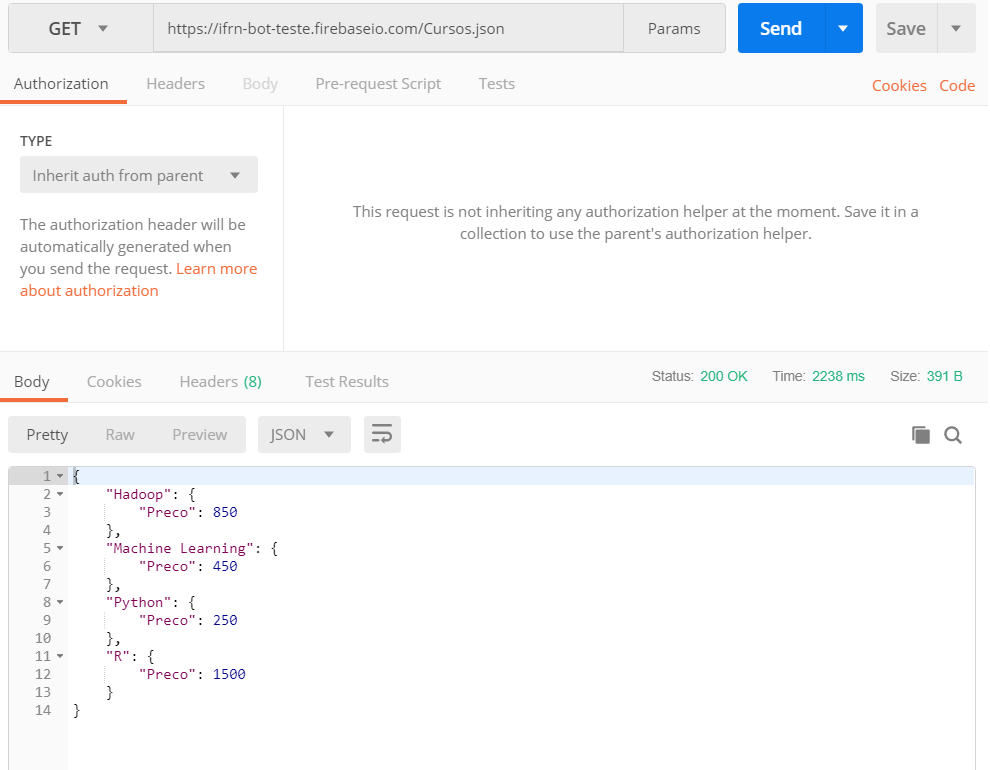


Figura 29.

Você também pode testar via Postman. Veja a Figura 30.



**Recuperando valores do banco de dados**

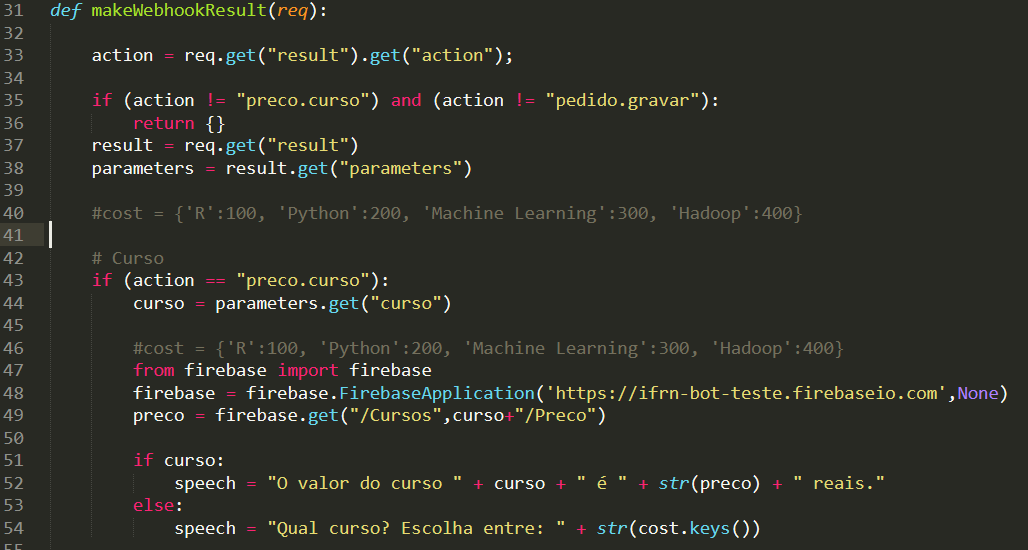
Antes de tudo, é necessário instalar localmente na sua máquina o pacote python-firebase e depois instalá-lo no heroku.

Então, abra um terminal e execute o comando

**pip install python-firebase** //Instala na máquina local

após instalação do pacote python-firebase, vamos fazer as alterações no backend. Observe que, na linha 6 fazemos a importação do pacote firebase e da linha 49 até a linha 51, estamos fazendo a leitura dos dados do banco de dados “*Realtime Database*”





Faça o deploy para o heroku

**git add .**

**git commit -am “firebase”**

**git push heroku master**

Teste no DialogFlow e no Postman

Precisamos ainda instalar o firebase no heroku.

**Instalando pacotes Python no Heroku**

Abra o arquivo requiments.txt na pasta onde está seu arquivo app.py e, acrescente o código da linha 2, como mostra a Figura 31.

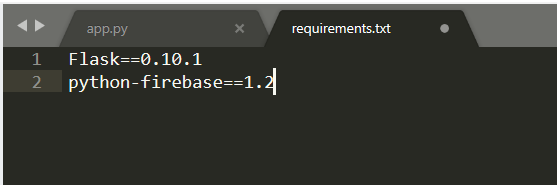


Figura 31. Instalando pacote Python no Heroku.

Faça novamente o deploy e teste.

Testando no Postman, Figura 32.

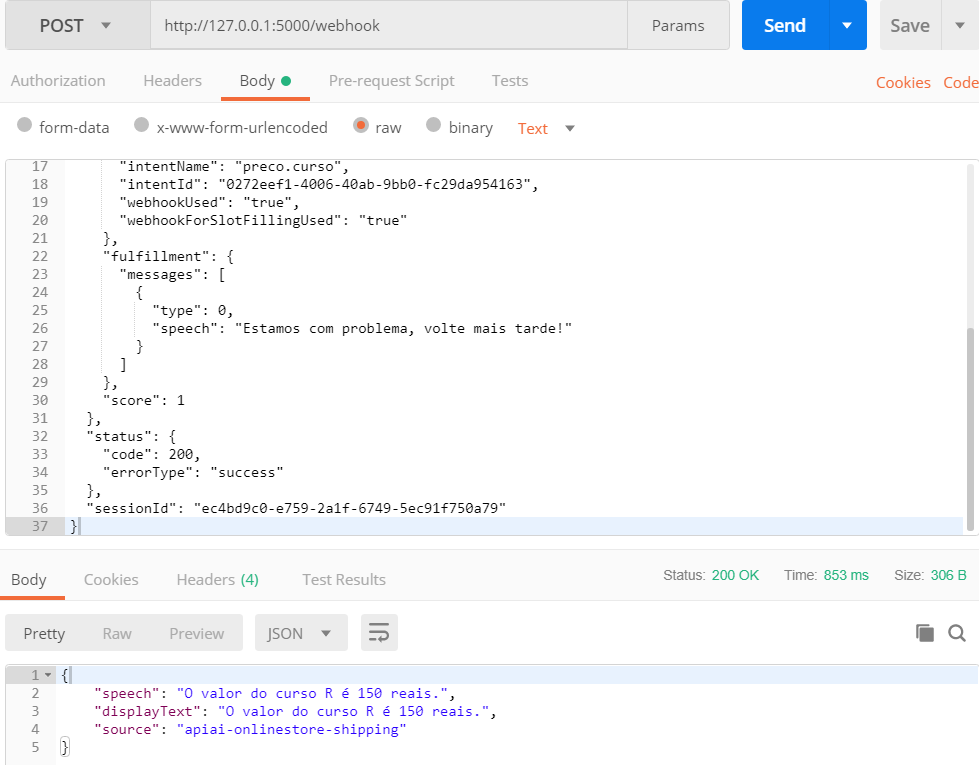


Figura 32.

Testando no DialogFlow, Figura 33.

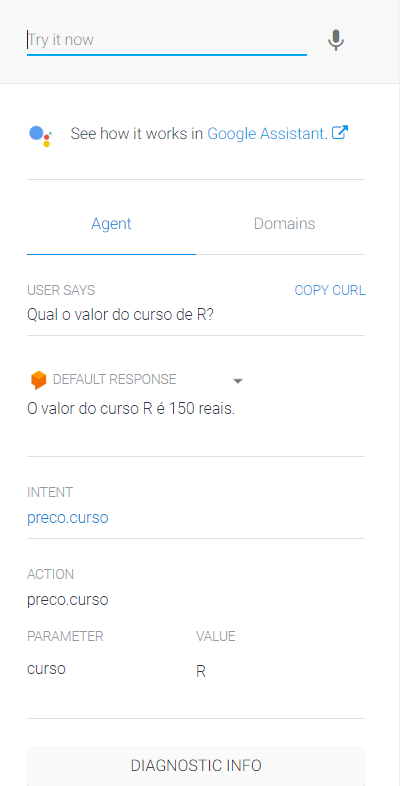


Figura 33.

Na seção anterior, fizemos apenas a leitura dos dados do banco de dados, que não precisa fazer autenticação. Agora vamos fazer a gravação de dados, onde é necessário se autenticar no banco de dados.

**Gravando dados no banco de dados**

Você deve criar uma intenção pedido.confirmacao-sim. Nessa intenção defina a ação (action) como pedido.gravar. Uma resposta do tipo “Estamos com problema, volte mais tarde!” e habilite essa intenção para usar WebHook.

Altere o código no backend para gravar o pedido.

