

# Actions on Google

Desenvolvendo actions para o Google Assistant com  **Kotlin** client library

# Wagner Messias



## Sobre mim...

- Desenvolvedor Android na Zup
- Bacharel em Sistemas de Informação
- Entusiasta da cultura Makers (Do It Yourself)

 @WagnerMessiasC

 wagnermessias

# Agenda

- Assistentes Virtuais
- Google Assistant
- Actions on Google
  - O que é Actions on Google?
  - Processo de desenvolvimento de actions
  - Recursos disponíveis
- Kotlin client library



Assistentes Virtuais



Ficção

Realidade

# Grande mudança na computação

## A cada 10 anos

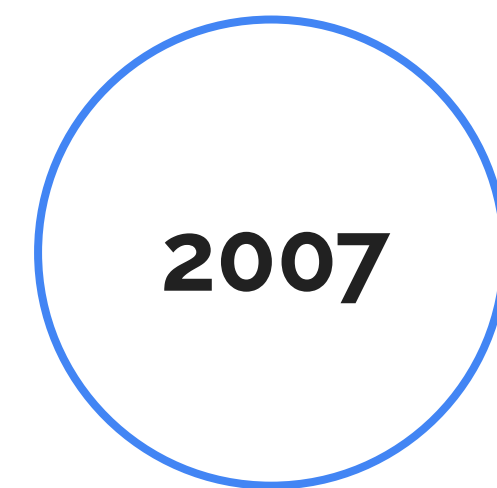
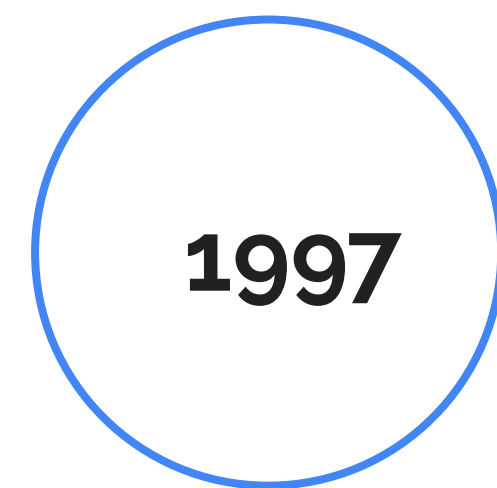
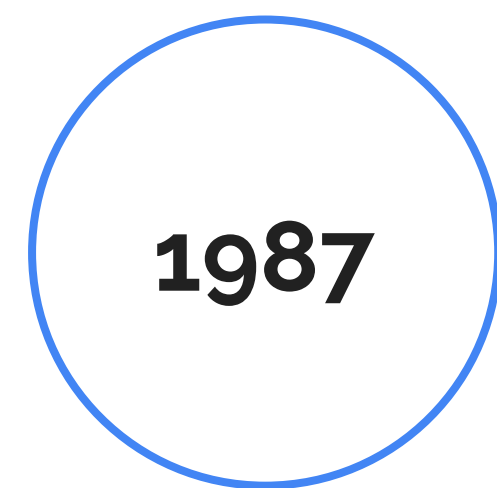
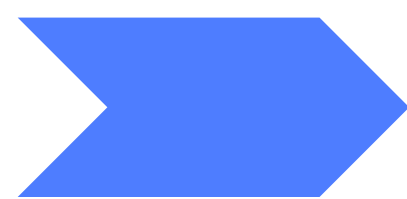
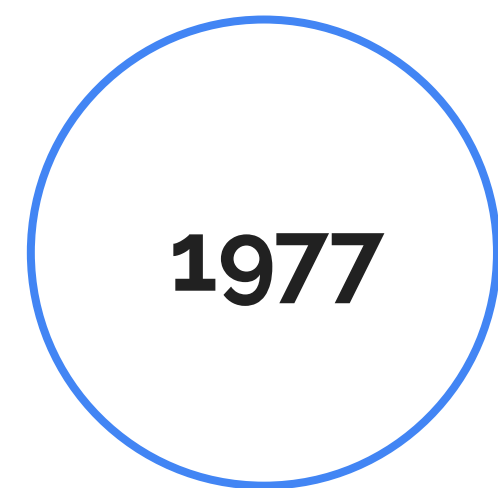
Mainframes

Desktop

Internet

Mobile

AI/Assistants



E a próxima mudança?



# Expectativas

- Principal Interface para Internet das coisas (IoT).
- Ajudar as pessoas em sua rotinas diárias.
- Atender necessidades dos usuários de forma mais rápida e intuitiva por voz.



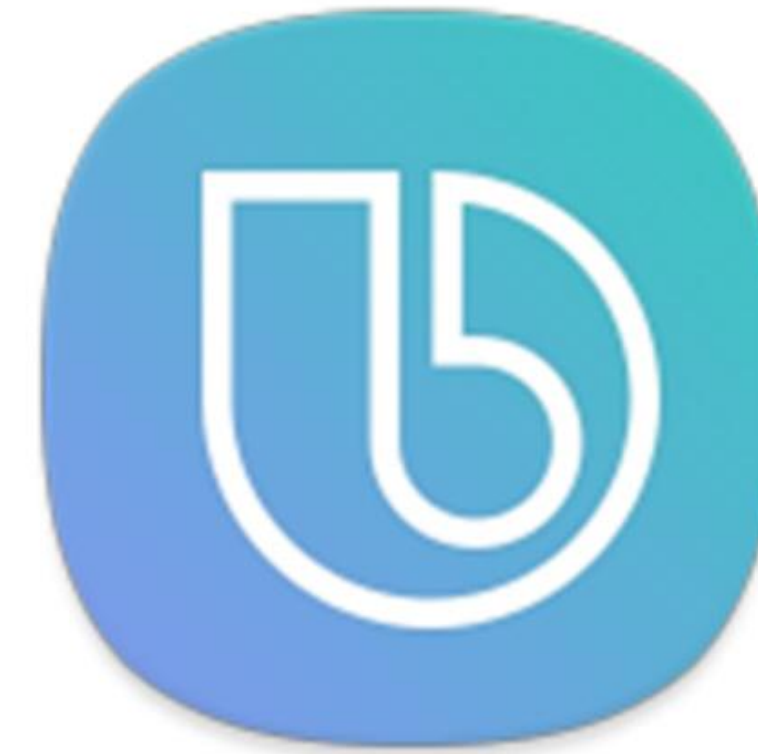
# Principais Assistentes Virtuais



Google Assistant



alexa



**Bixby**



**Siri**



**Cortana**



WESTGATE



Hey Google

Google Assistant

Hi how can I help?

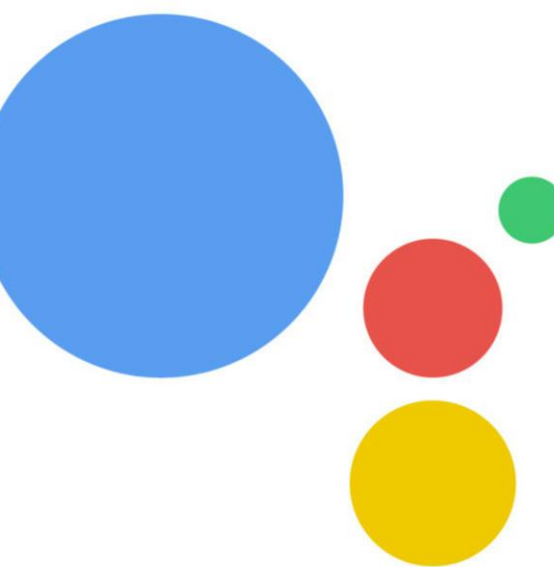
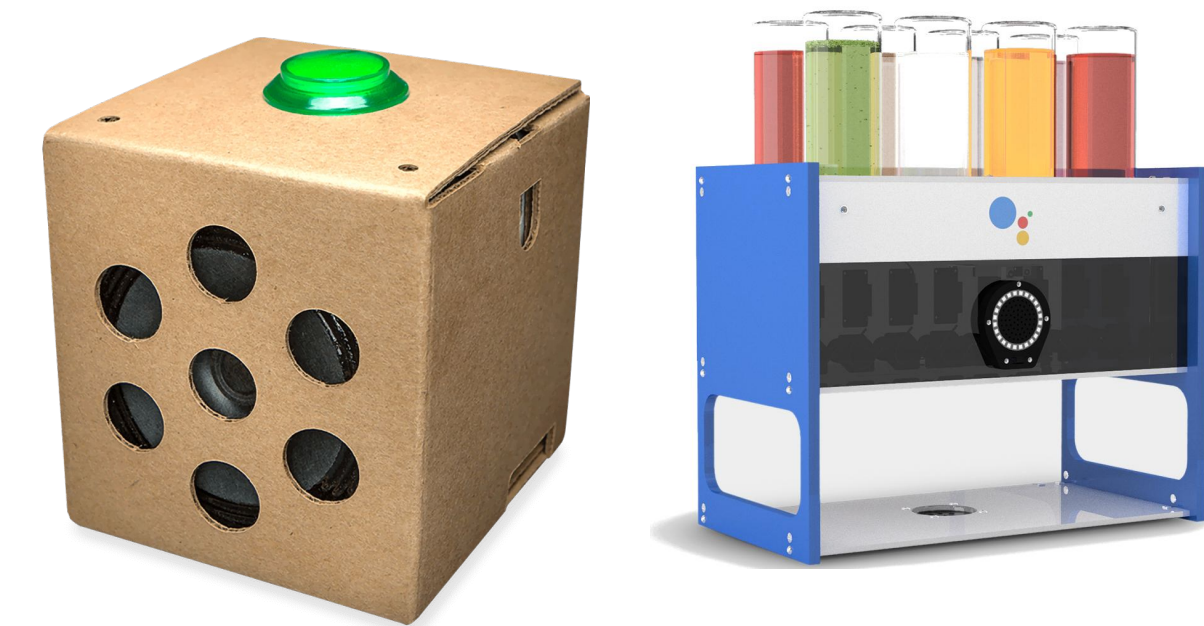
## Suporte crescente de idiomas



1B+ Dispositivos



SDK



Bilíngue

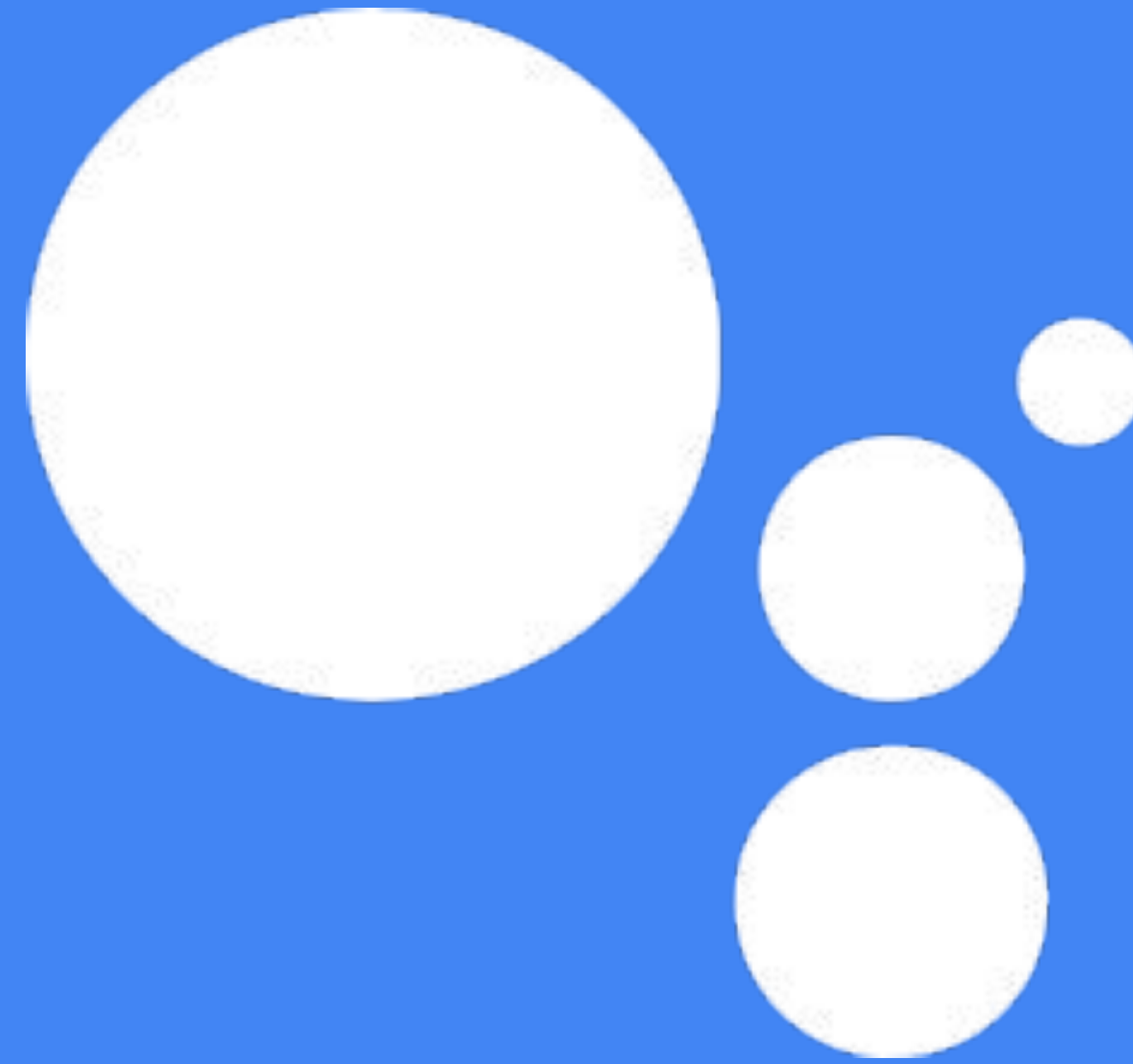


# Home Devices



**Mas, o que posso desenvolver? e como?**

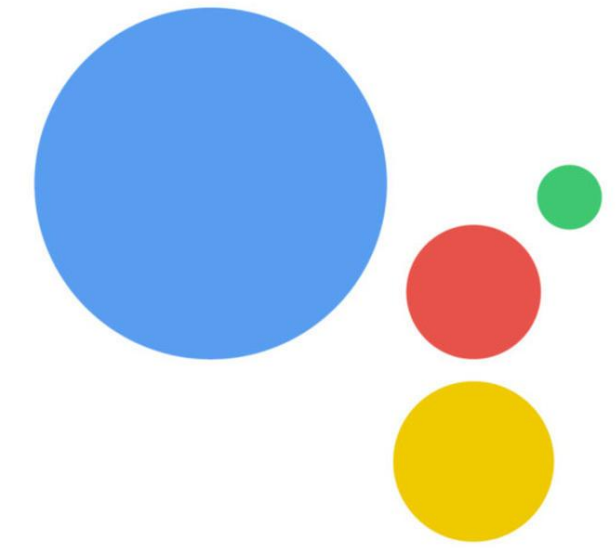




# Actions on Google

Desenvolvimento de aplicações de voz

# O que é Actions on Google?



É uma plataforma que viabiliza o desenvolvimento de “Assistant Apps”, aplicativos para expandirem as funcionalidades do Google Assistant para serviços de terceiros, através de actions.

# O que é Actions on Google?



Actions on Google

Vertical Solutions

Docs

Community P >



Pesquisa

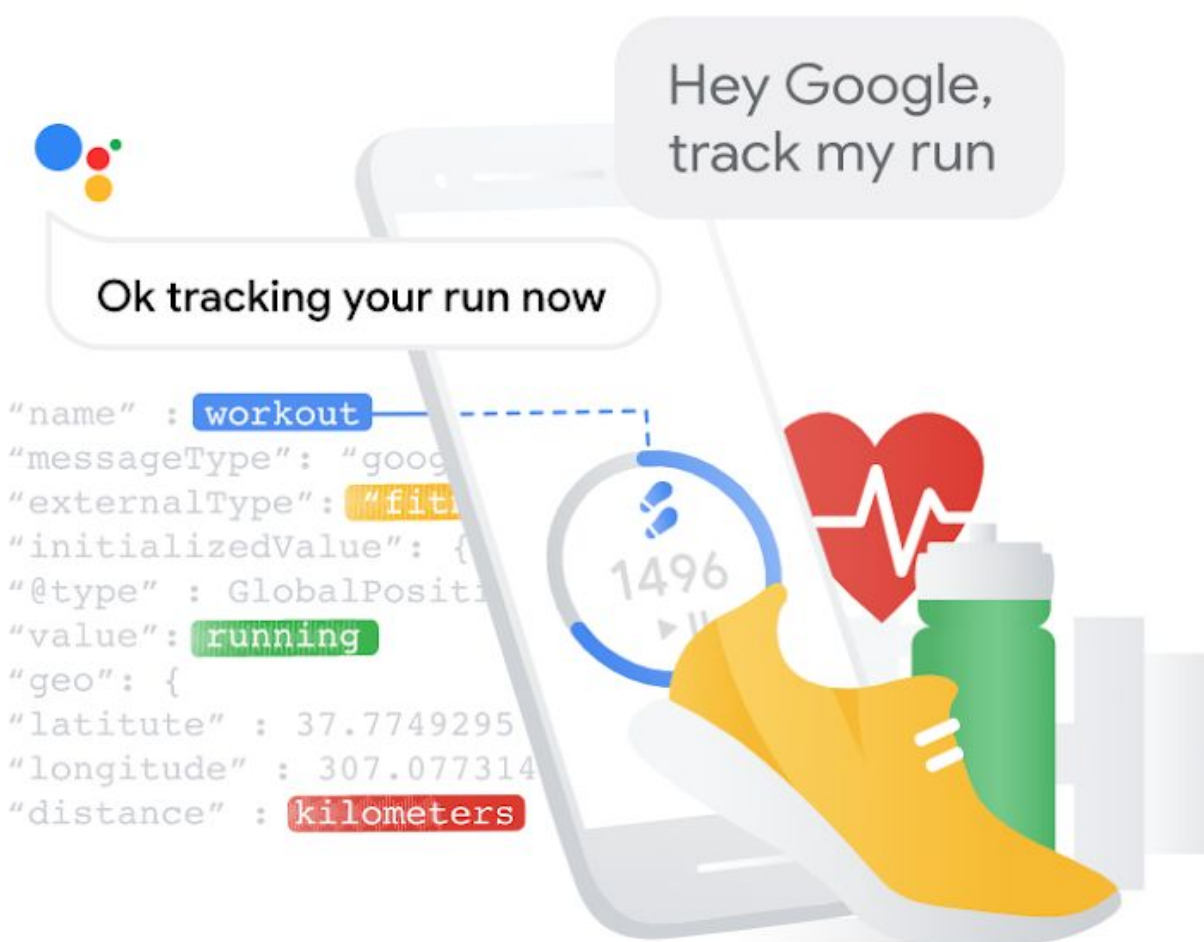
LANGUAGE ▾

TODOS OS PRODUTOS ⋮



Build Actions to help users get things done with the Google Assistant.

[GO TO ACTIONS CONSOLE](#)



## Build Actions to help users get things done with the Google Assistant

With Actions on Google, you can easily reach and engage with users across Google. From quick commands to full conversations, Actions help you connect your content and services to users with the Google Assistant.

[GET STARTED](#)

### Build easily

Leverage Google's capabilities in machine learning and Natural Language Understanding (NLU) to easily design conversations and deploy Actions.

### Appear everywhere

The Google Assistant is available across 1B+ devices like speakers, smart displays, phones, and more. Build for voice, visuals, or both.

### Grow usage

Increase discovery and usage of your Actions with features like Action Links, notifications, and localization to nearly 30 languages and 80 countries.



# O que é Action?

**É o ponto de entrada em uma interação entre o usuário e o Google Assistant, onde usuário pode falar ou digitar uma frase informando o nome da action que deseja interagir.**

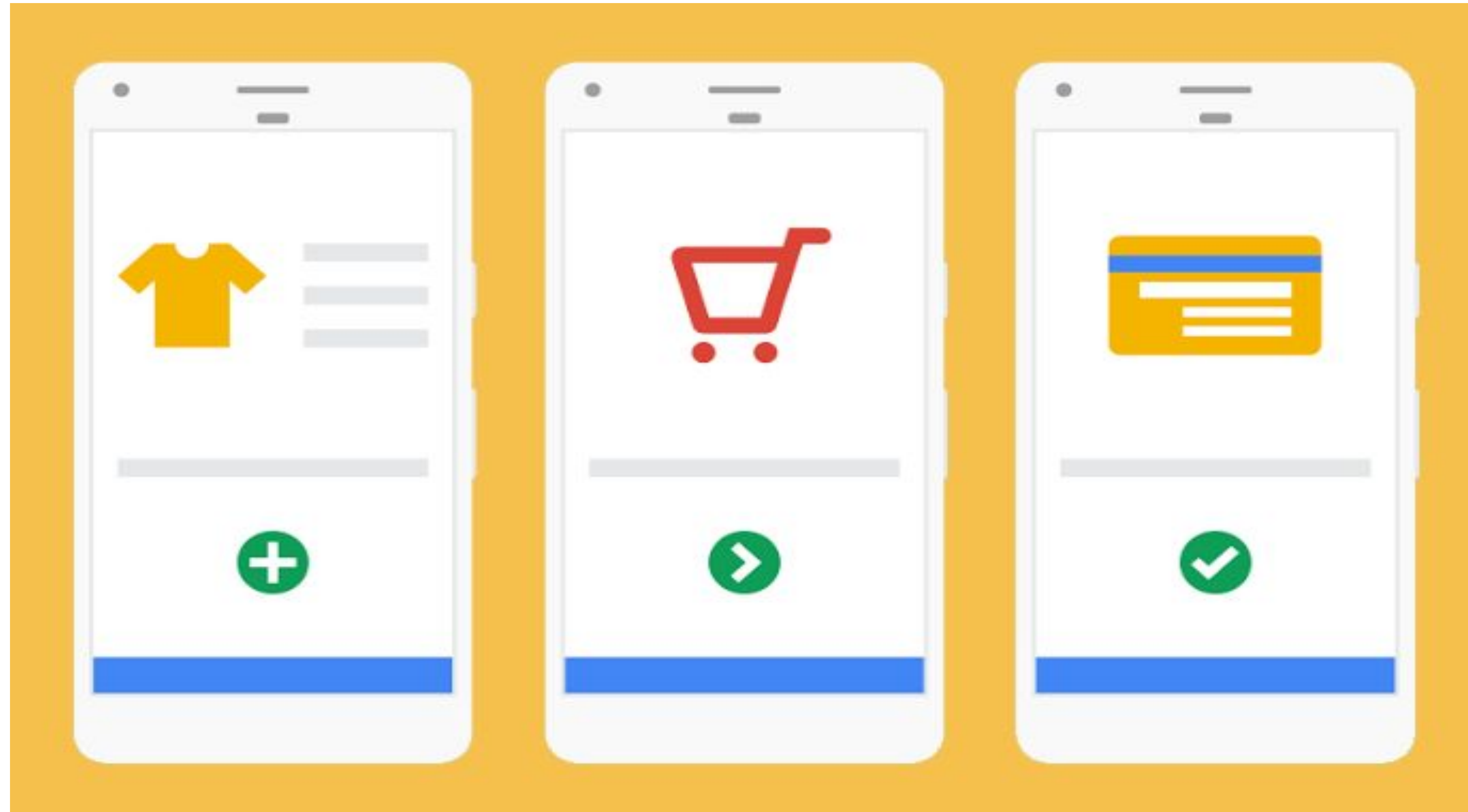
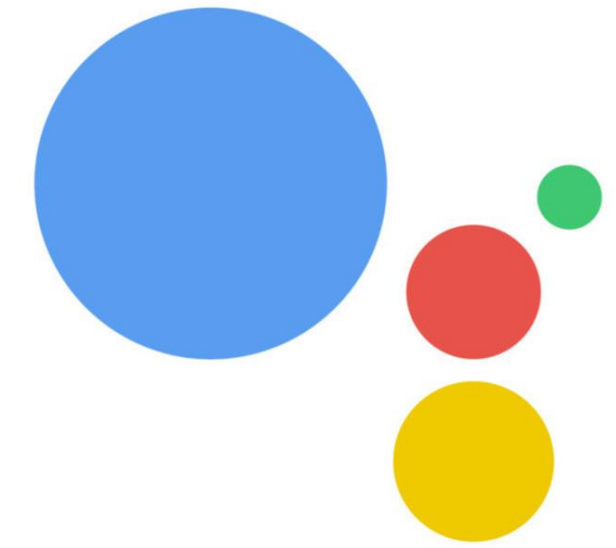
# Invocação

Ok Google, falar com Pizzaria do Gordo

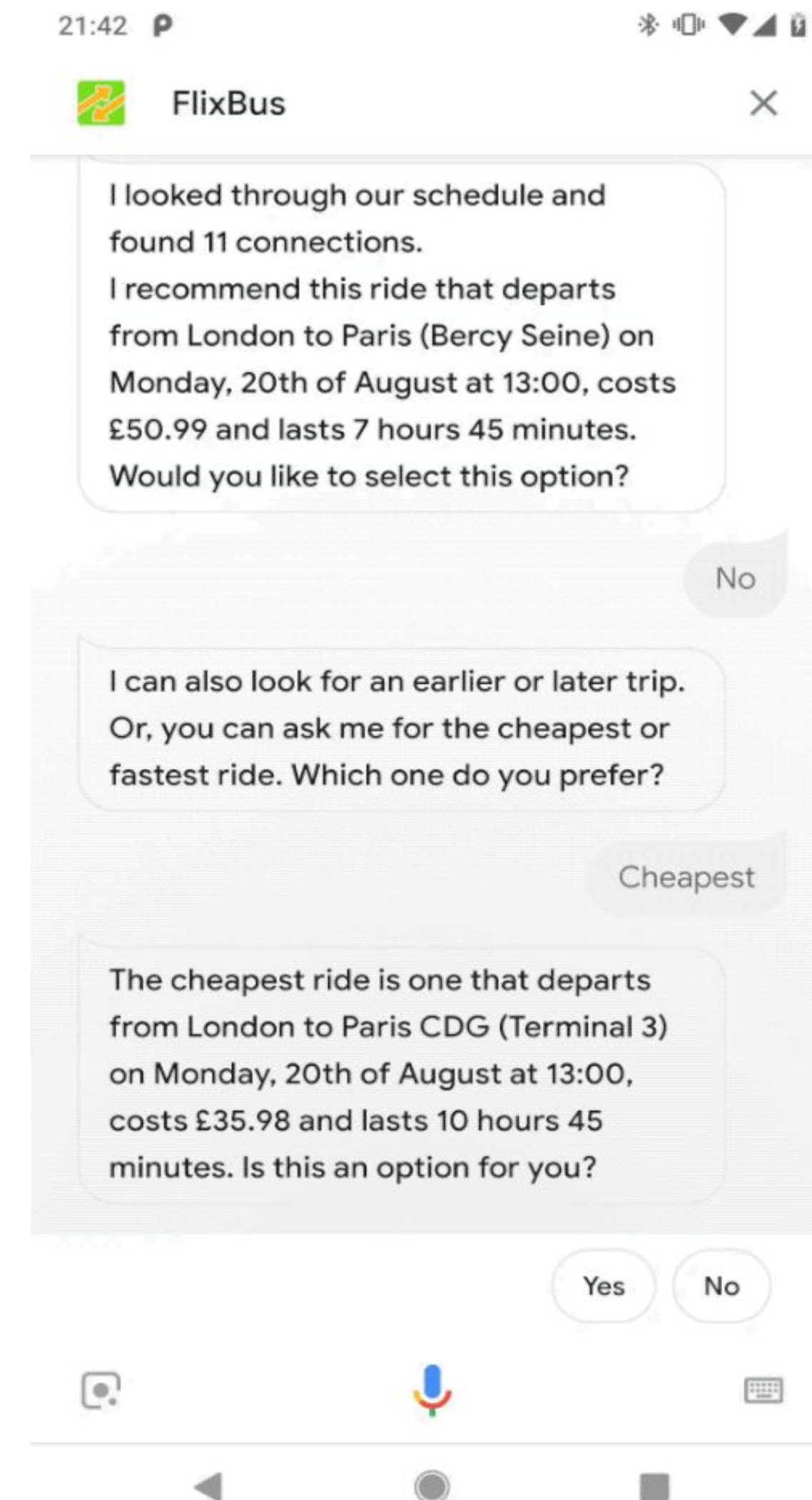
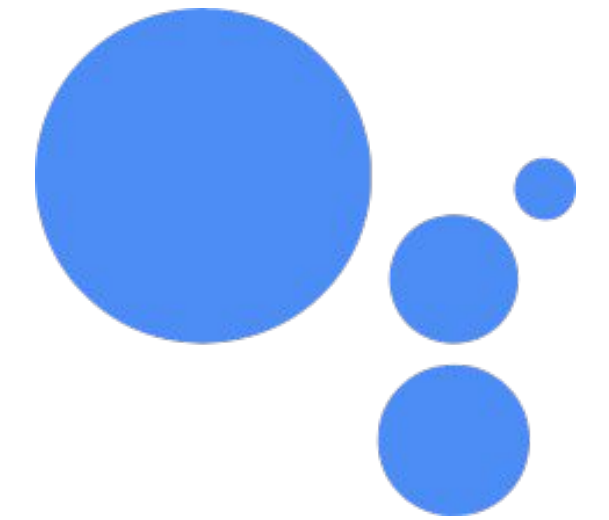
↑  
**Trigger Phrase**

↑  
**Action**  
**(Especificada pelo desenvolvedor)**

# Venda de bens e serviços físicos



# Uso do **Google Pay** para finalizar as transações.



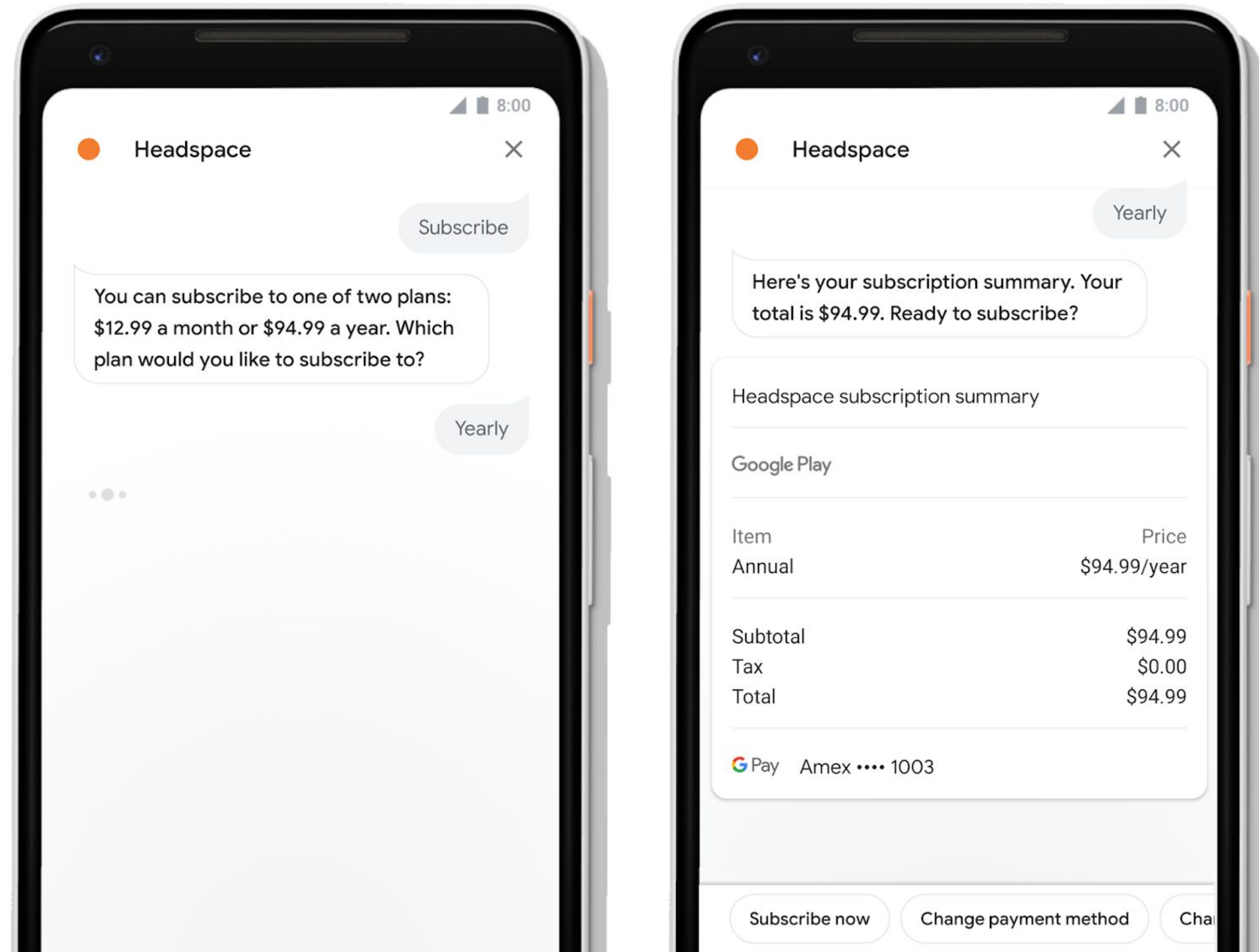
Quem quer rir  
tem que fazer  
rir.



**VOCÊ TEM QUE ME AJUDAR A TE AJUDAR...**

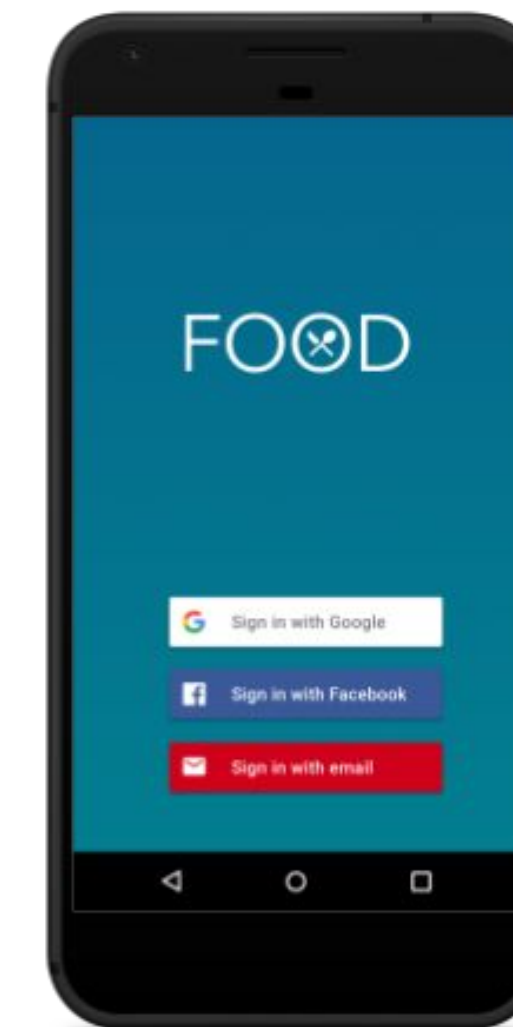
# Venda de produtos **Digitais**

Incluindo compras únicas como atualizações - pacotes de expansão ou novos níveis, por exemplo - e até mesmo assinaturas recorrentes diretamente sua **Action**.



# Autenticação

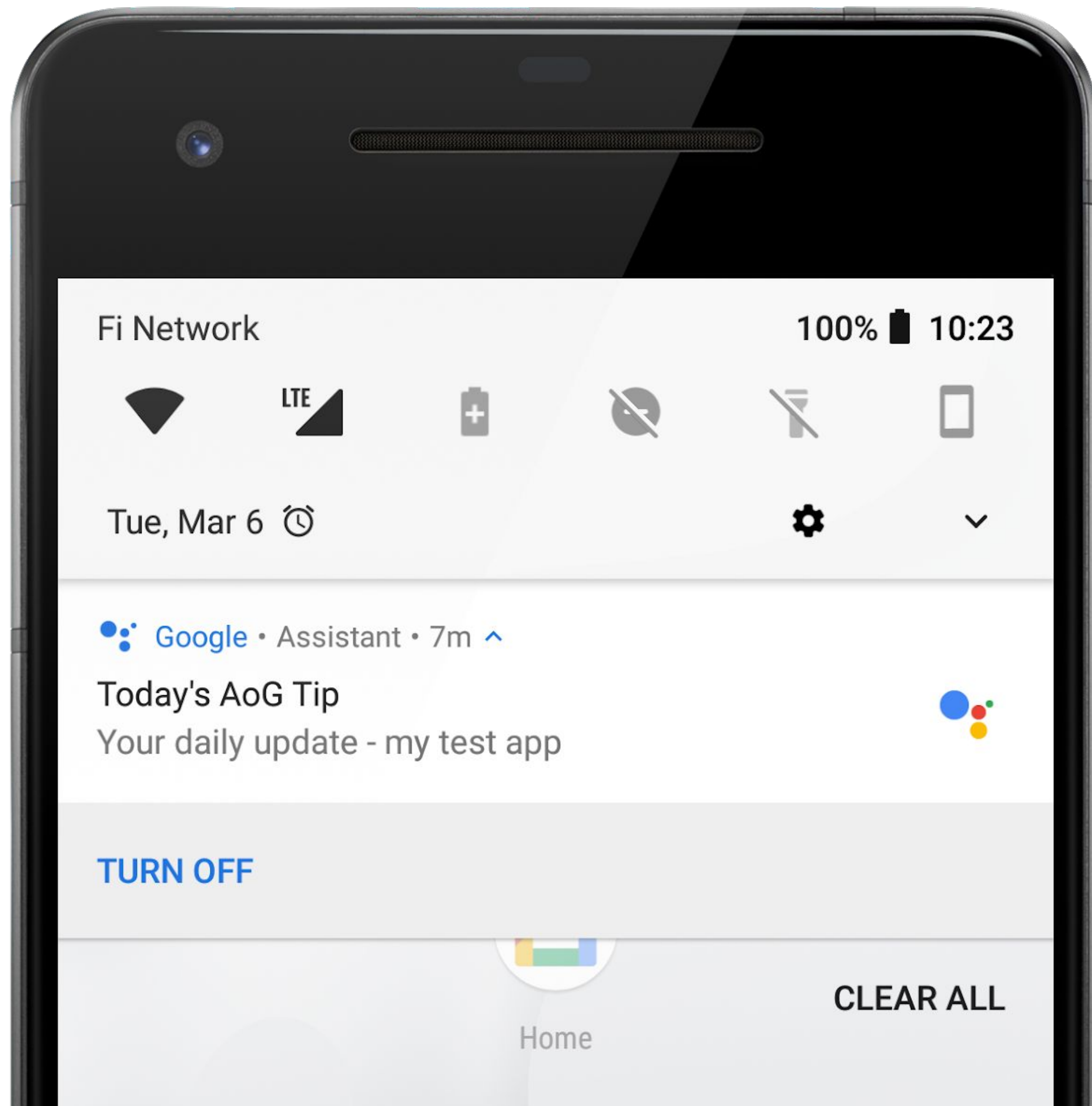
- Google Sign-in
- OAuth Google Sign-in
- OAuth (Não recomendado)





# Envolver os usuários

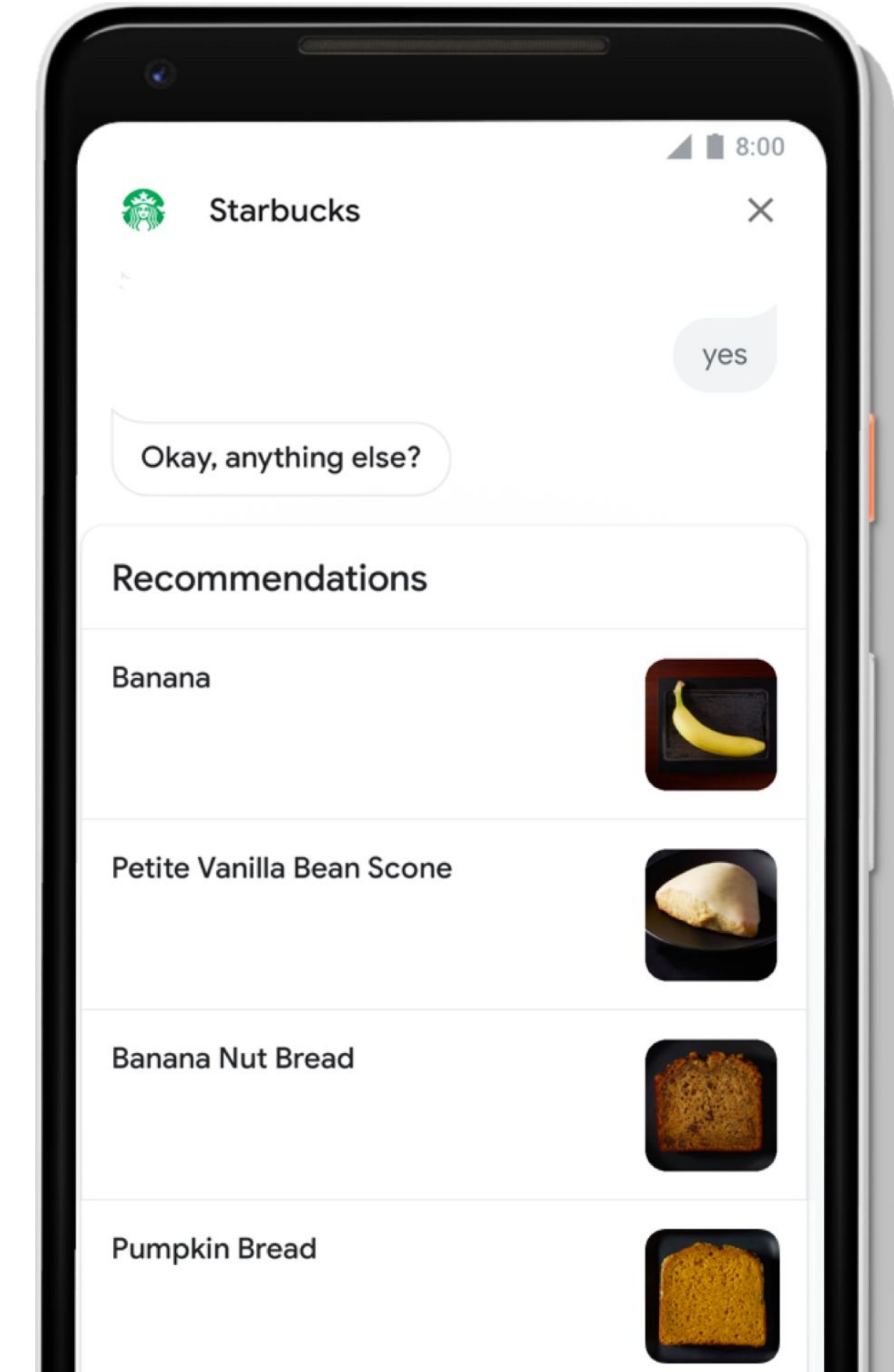
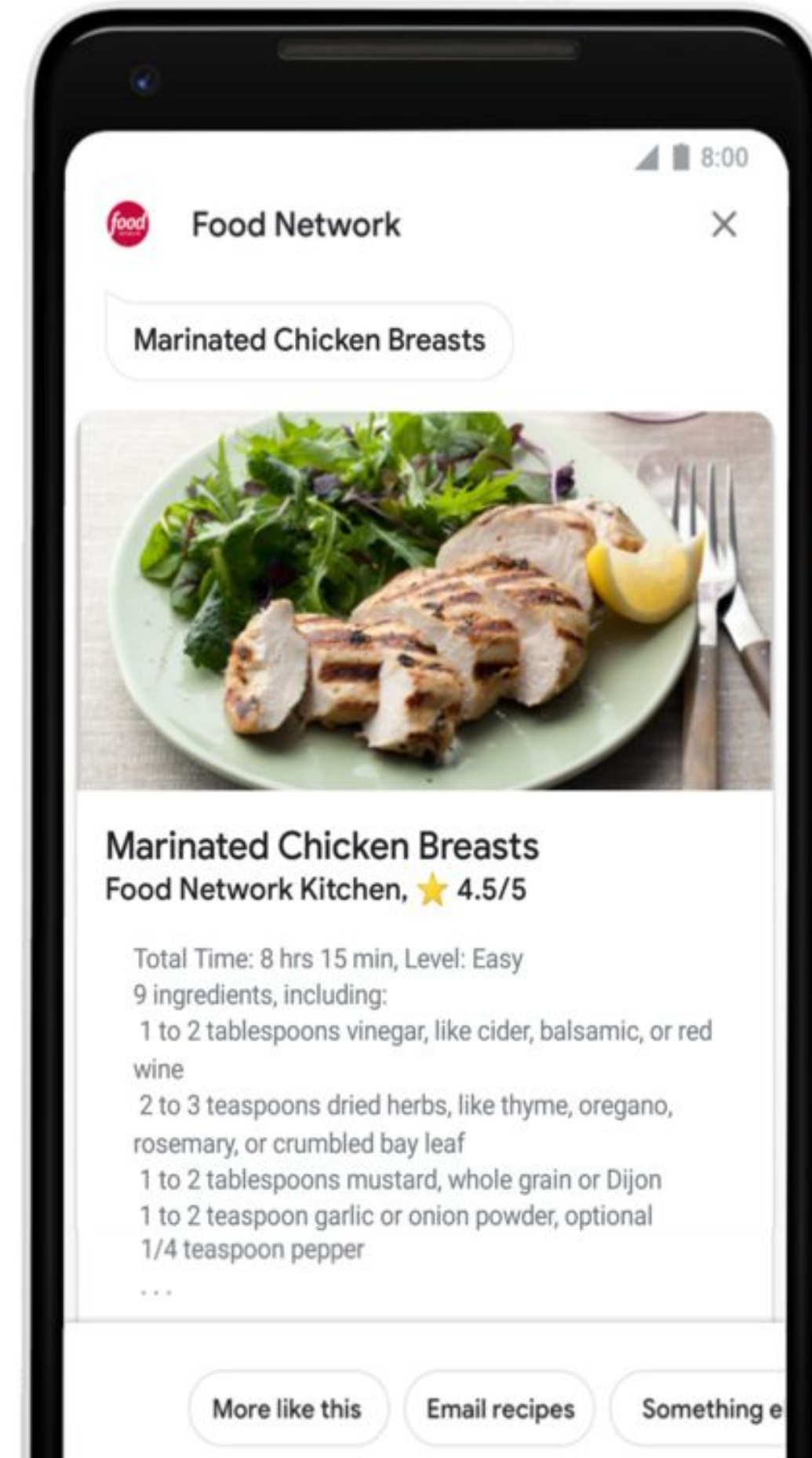
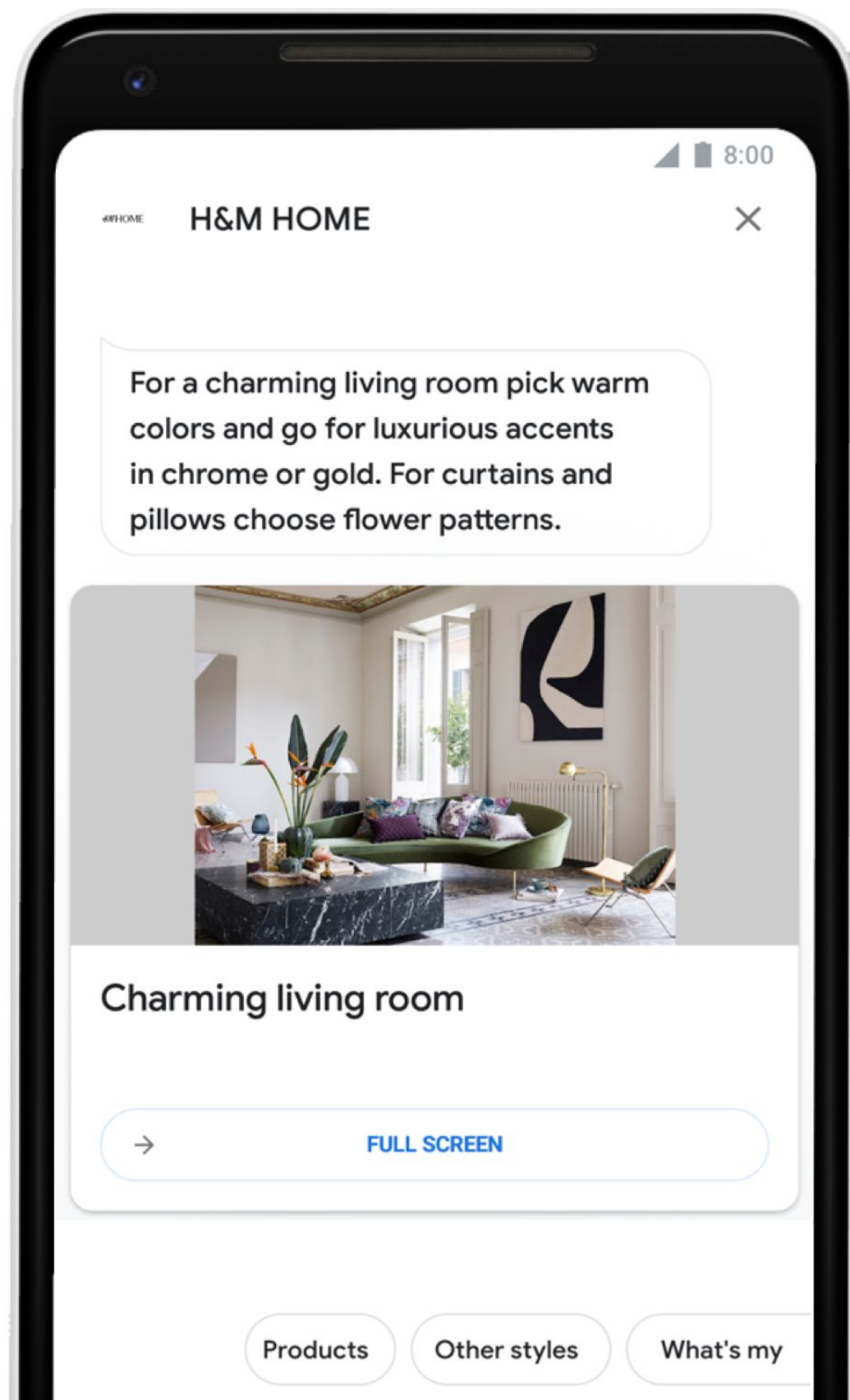
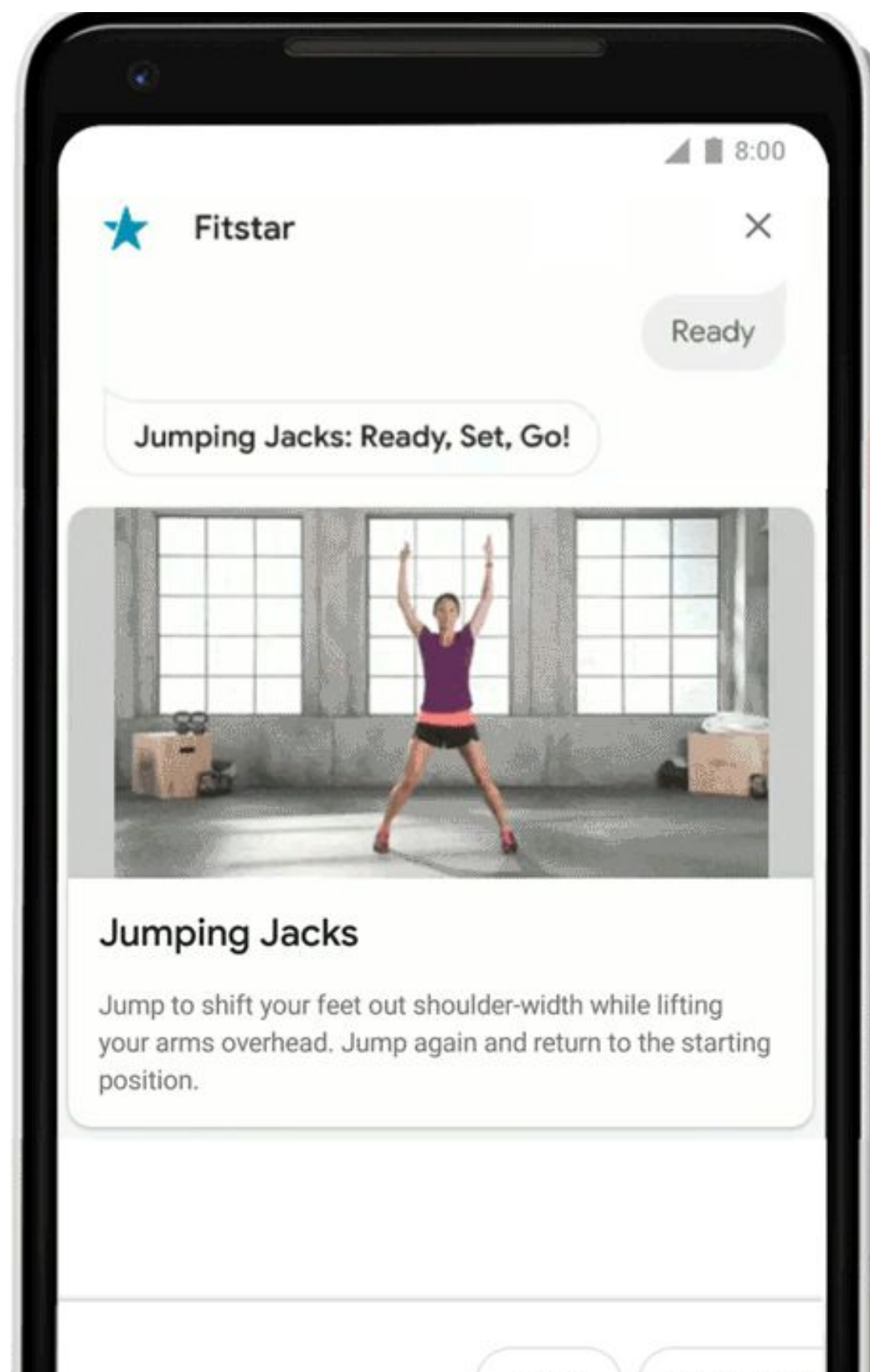
Push notifications



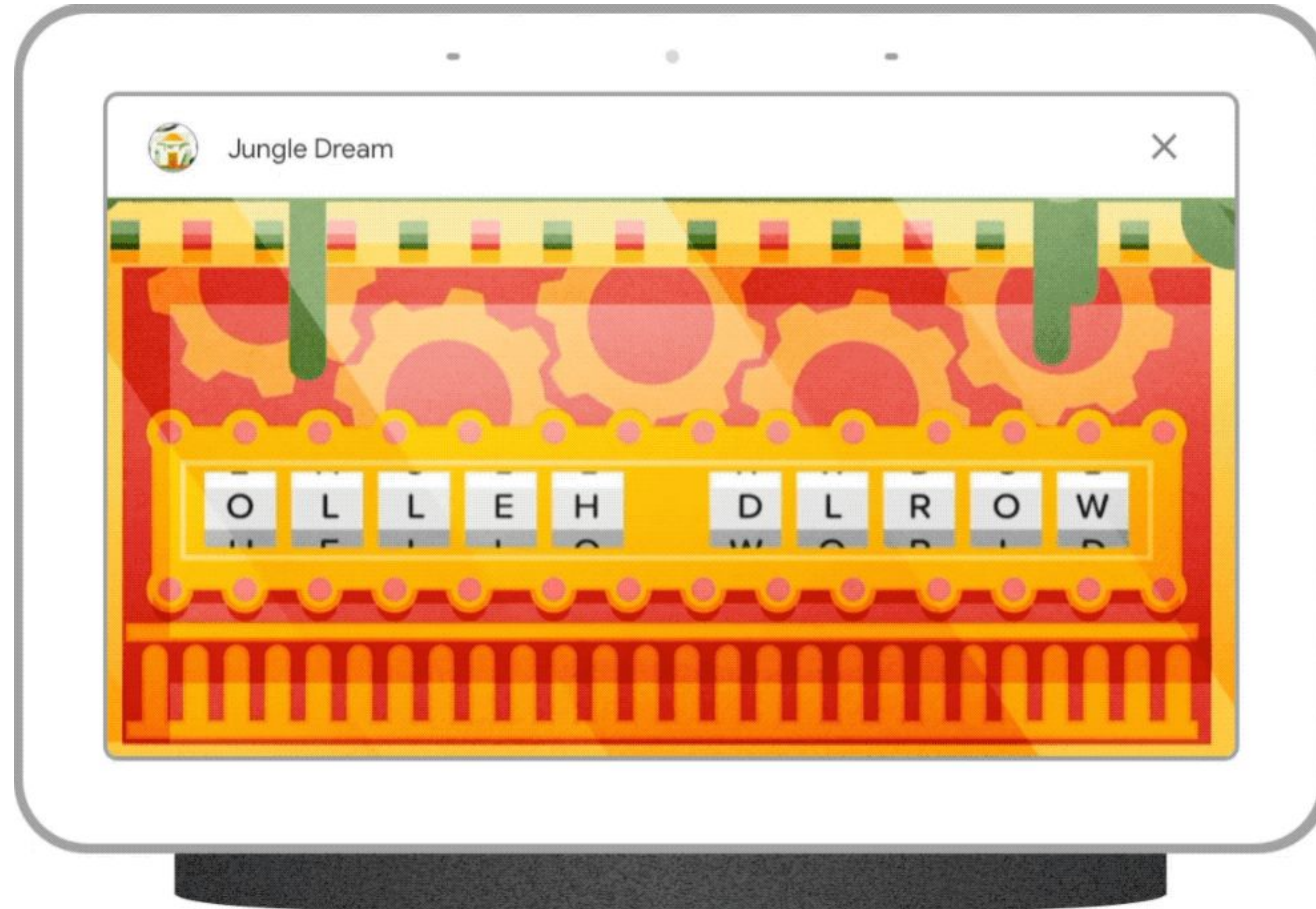
Links para Action



# Resposta simples e ricas



# Interactive Canvas (GAMES)



# Google Assistant no Brasil



Magazine Luiza

Pergunte ao assistente o status do seu último pedido realizado no site ou aplicativo Magalu através de um comando de voz. Você também pode



Conversar com Magazine Luiza

★ 4.1



BIA do Bradesco

A BIA (Bradesco Inteligência Artificial) do Bradesco é uma entidade que usa inteligência artificial para ajudar clientes do Bradesco e o público em



Conversar com a BIA do Bradesco

★ 4.5



Caixa

O Assistente Virtual Caixa fornece o resultado das Loterias, informa a data e estimativa dos próximos sorteio para todas as modalidades, permite...



Falar com a Caixa

★ 4.7



Galinha Pintadinha

Fale com a Galinha Pintadinha e divirta-se com brincadeiras, músicas e vídeos. A Popó está ansiosa para conversar com você e está te esperando.



Falar com a Galinha Pintadinha

★ 4.7



G1 Eleições

O Assistente possui diversas informações sobre eleições, incluindo as mais atuais notícias, informações e opiniões dos candidatos, boatos e



Falar com G1 Eleições

★ 4.1

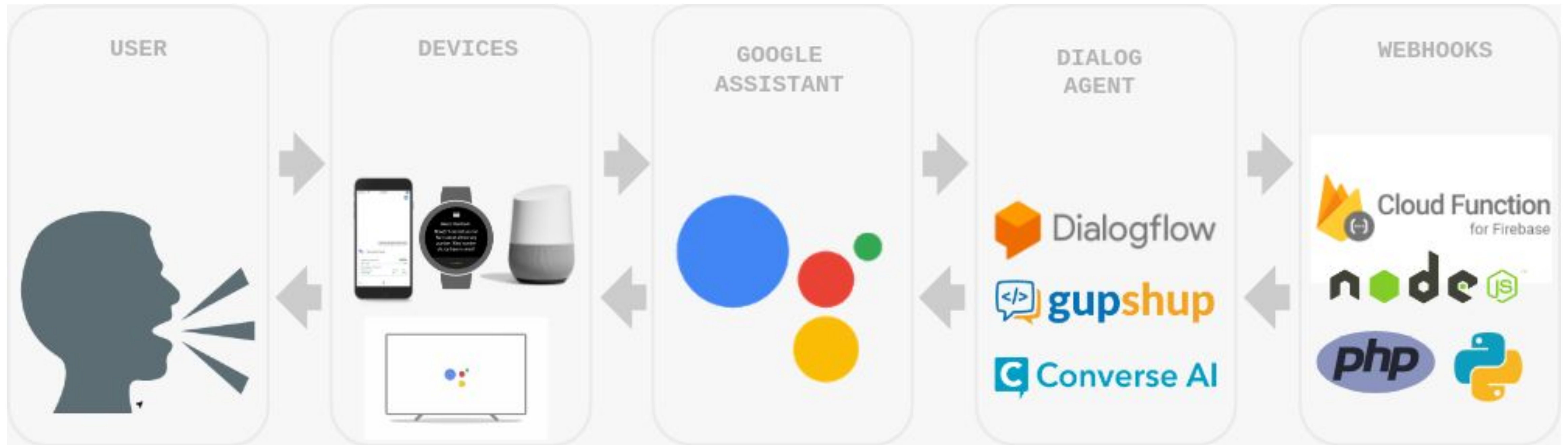
# Actions Console

The screenshot displays the Actions Console interface. At the top, there is a purple header bar with the "Actions on Google" logo on the left, the user name "Contando os Dias" in the center, and "Go to docs" and a profile icon on the right. Below the header is a navigation sidebar on the left, divided into sections: "Overview" (selected), "SETUP" (containing "Invocation"), "BUILD" (containing "Actions" and "Theme customization"), "TEST" (containing "Simulator"), "DEPLOY" (containing "Directory information", "Location targeting", "Surface capabilities", and "Release"), "MEASURE" (containing "Analytics"), and "ADVANCED OPTIONS" (containing "Account linking" and "Brand verification"). The main content area is titled "Overview" and features a welcome message: "Welcome to your project! This is where you'll be managing your Actions." Below this, there are four expandable cards: "Portuguese (Brazil)" (with a "Modify languages" link), "Quick setup" (with a "Decide how your Action is invoked" link), "Build your Action", and "Get ready for deployment". At the bottom, there is a "Release" card with a checkmark and the text "You have not created a release for the current draft." A back arrow is visible at the bottom left of the sidebar.

# Actions Console

- Adicionar ou importar projetos de actions
- Gerenciamento e personalização das actions
- Simulador para testar suas Actions
- Gerenciar release (Alpha, Beta e Production)
- Deploy do projeto
- Analytics
- Integração com Backend Services
- Entre outros...

# Fluxo de interação de uma Action



# Agente **Dialogflow** (api.ai)



Poderoso mecanismo de entendimento de linguagem natural (NLU) para processar e entender a entrada de linguagem natural.



# Agente Dialogflow (api.ai)

## pedido-pizza (Intent)

“Gostaria de uma **mussarela** **grande** e uma **fanta uva** de **2 litros**”

**@sabor** (obrigatório)

**@tamanho**(obrigatório)

**@bebida**

**@bebida-qtd**(obrigatório / se conter bebida)

# Fulfillment - Inline Editor

The screenshot shows the Dialogflow Fulfillment Inline Editor interface. On the left is a navigation sidebar with the Dialogflow logo and menu items: Concrete (owned by yp...), Intents, Entities, Knowledge [beta], Fulfillment (highlighted), Integrations, Training, History, Analytics, Prebuilt Agents, Small Talk, and Docs. The main content area is titled 'Fulfillment' and contains two sections: 'Webhook' (DISABLED) and 'Inline Editor (Powered by Cloud Functions for Firebase)' (ENABLED). The Inline Editor section includes a description, a 'Docs' link, and a code editor for 'index.js'. The code editor shows JavaScript code for setting up a fulfillment function. At the bottom, there is a 'View execution logs in the Firebase console' link, a timestamp 'Last deployed on 02/26/2019 10:55', and a blue 'DEPLOY' button.

**Dialogflow**

Concrete (owned by yp...)

en pt-BR

Intents

Entities

Knowledge [beta]

**Fulfillment**

Integrations

Training

History

Analytics

Prebuilt Agents

Small Talk

Docs

**Fulfillment**

**Webhook** DISABLED

Your web service will receive a POST request from Dialogflow in the form of the response to a user query matched by intents with webhook enabled. Be sure that your web service meets all the [webhook requirements](#) specific to the API version enabled in this agent.

**Inline Editor** (Powered by Cloud Functions for Firebase) ENABLED

Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. [Docs](#)

index.js package.json

```
1 'use strict';
2
3 const {
4   dialogflow,
5   BasicCard,
6   Image,
7   Button,
8   Permission
9 } = require('actions-on-google');
10 const functions = require('firebase-functions');
11
12 const app = dialogflow({debug: true});
13
14 app.intent('concrete.history', (conv) => {
15   const card = new BasicCard({
16     title: 'Quem somos nós',
```

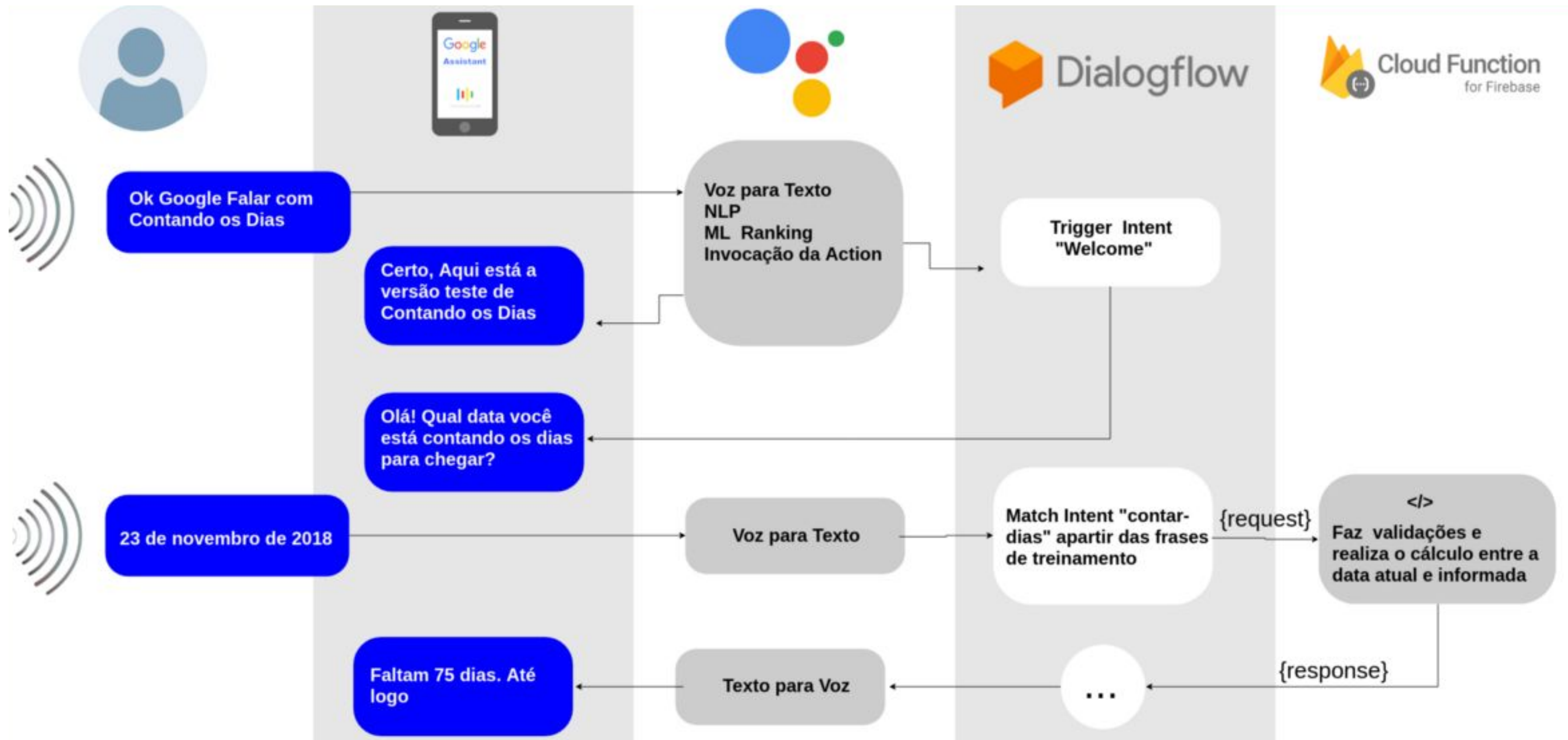
[View execution logs in the Firebase console](#) Last deployed on 02/26/2019 10:55 **DEPLOY**

# Fulfillment - Webhook

The screenshot shows the Dialogflow Fulfillment configuration page. On the left is a sidebar with the Dialogflow logo and navigation options: Concrete (owned by yp...), language tabs for 'en' and 'pt-BR', Intents, Entities, Knowledge (beta), Fulfillment (highlighted), Integrations, Training, History, Analytics, Prebuilt Agents, Small Talk, and Docs. The main content area is titled 'Fulfillment' and contains a 'Webhook' section. The Webhook is currently 'ENABLED'. Below this, there is a text box for the URL, set to 'https://us-central1-concrete-cfaec.cloudfunctions.net/dialogflowFirebaseFulfillment'. The 'BASIC AUTH' section has input fields for 'Enter username' and 'Enter password'. The 'HEADERS' section has input fields for 'Enter key' and 'Enter value', along with an 'Add header' button. The 'DOMAINS' section has a dropdown menu set to 'Disable webhook for all domains'. Below the Webhook section is an 'Inline Editor' section, which is currently 'DISABLED'. It includes a description: 'Build and manage fulfillment directly in Dialogflow via Cloud Functions for Firebase. Docs'. At the bottom, there is a code editor showing the start of an 'index.js' file with the following code:

```
index.js package.json
1 'use strict';
2
3 const {
4   dialogflow,
5   BasicCard,
```

# Fluxo Aplicação de Voz

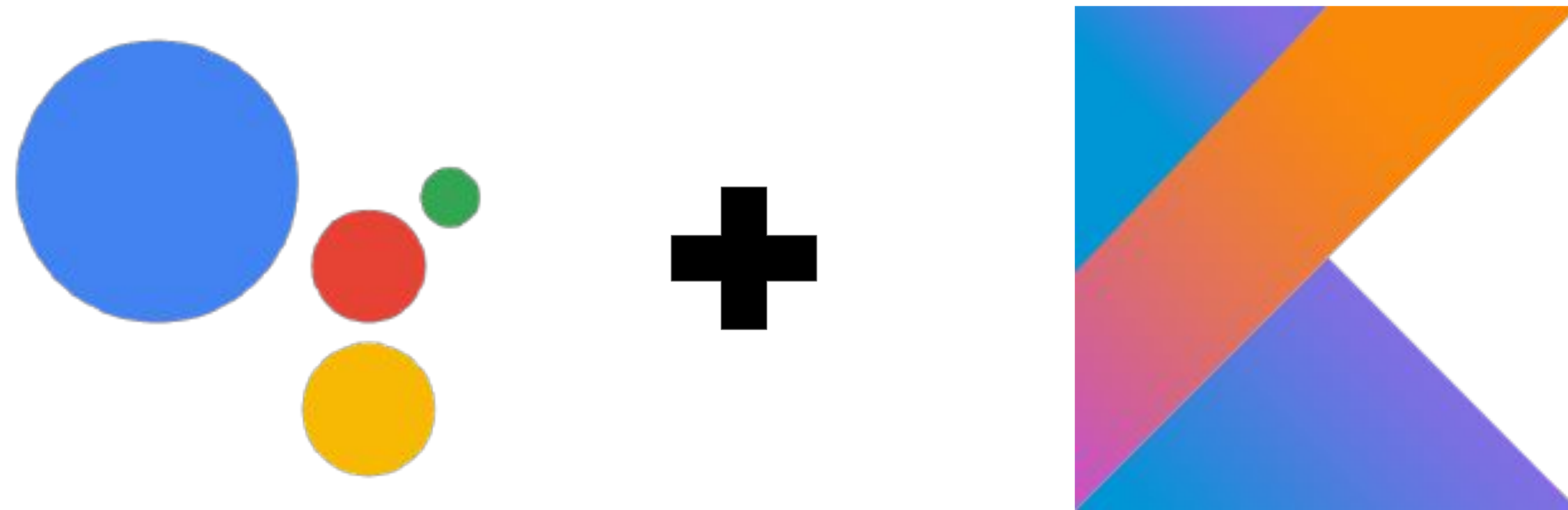


# Request

```
29  [-] {
30      "responseId": "a89ed7ec-5259-46f2-bee4-8a2858f70ba7-fd8ff490",
31  [-] "queryResult": {
32      "queryText": "GOOGLE_ASSISTANT_WELCOME",
33      "action": "input.welcome",
34      "parameters": {},
35      "allRequiredParamsPresent": true,
36  [+  "fulfillmentMessages": [...
44      ],
45  [+  "outputContexts": [...
73      ],
74  [-] "intent": {
75      "name": "projects/testes-8ee10/agent/intents/ec532c8f-2ddc-40b5-9757-31aa120a410d",
76      "displayName": "welcome"
77  },
78      "intentDetectionConfidence": 1,
79      "languageCode": "pt-br"
80  },
81  [+  "originalDetectIntentRequest": {...
132  },
133      "session": "projects/testes-8ee10/agent/sessions/1563459733813662119458"
134  }
```

# Response

```
1  {
2  "payload": {
3    "google": {
4      "expectUserResponse": true,
5      "richResponse": {
6        "items": [
7          {
8            "simpleResponse": {
9              "textToSpeech": "Bem-vindo, ao canvas teste"
10           }
11         },
12         {
13           "immersiveResponse": {
14             "loadImmersiveUrl": "https://canvasactions.herokuapp.com"
15           }
16         }
17       ]
18     }
19   }
20 }
21 }
```



**Kotlin Client Library**  
**para [Actions on Google](#)**

# Requisitos

- JDK 8 ou superior.
- Pode ser Instalado usando Gradle e Maven
- Pode ser usada em conjunto com Dialogflow para integração com Actions on Google ou Actions SDK



# Setup

```
1 repositories {
2     mavenCentral()
3 }
4
5 dependencies {
6     compile group: 'com.google.actions', name: 'actions-on-google', version: '1.3.0'
7 }
```

# Class DialogflowApp

```
class MyAction: DialogflowApp() {  
    // Todo sua classe App  
}
```

# Handle Request

```
fun main(args: Array<String>): Unit = io.ktor.server.netty.EngineMain.main(args)
```

```
@kotlin.jvm.JvmOverloads
```

```
fun Application.module(testing: Boolean = false) {
```

```
    val actionApp: App = MyAction()
```

```
    install(ContentNegotiation) { this: ContentNegotiation.Configuration
```

```
        gson {}
```

```
    }
```

```
    routing { this: Routing
```

```
        post( path: "/mywebhook" ) { this: PipelineContext<Unit, ApplicationCall>
```

```
            val requestBodyObject = call.receiveOrNull<Any>()
```

```
            val gson = GsonBuilder().create() // for pretty print feature
```

```
            val bodyJson = gson.toJson(requestBodyObject)
```

```
            val headersMap = call.request.headers.toMap()
```

```
            call.respond(actionApp.handleRequest(bodyJson, headersMap).get())
```

```
        }
```

```
        get( path: "/" ) { this: PipelineContext<Unit, ApplicationCall>
```

```
            call.respondText( text: "running!", ContentType.Text.Plain)
```

```
        }
```

```
    }
```

```
}
```

# Handle Request

```
val actionApp: App = MyAction()
```

```
routing { this: Routing
  post( path: "/mywebhook") { this: PipelineContext<Unit, ApplicationCall>
    val requestBodyObject = call.receiveOrNull<Any>()
    val gson = GsonBuilder().create() // for pretty print feature
    val bodyJson = gson.toJson(requestBodyObject)
    val headersMap = call.request.headers.toMap()

    call.respond(actionApp.handleRequest(bodyJson, headersMap).get())
  }
}
```

# @ForIntent

```
class MyAction: DialogflowApp() {  
    @ForIntent(value: "default welcome")  
    fun welcome(request: ActionRequest): ActionResponse {  
        val responseBuilder = getResponseBuilder(request)  
        responseBuilder.add("Olá, qual seu pedido para hoje?")  
        return responseBuilder.build()  
    }  
}
```

# ActionRequest

```
@ForIntent( value: "pedido-pizza")
fun pedidoPizza(request: ActionRequest): ActionResponse {

    val responseBuilder = getResponseBuilder(request)

    val sabor = request.getParameter( name: "sabor")
    val tamanho = request.getParameter( name: "tamanho")
    val bebida = request.getParameter( name: "bebida")
    val quantidadeBebida = request.getParameter( name: "bebida-qtd")

    //Calcular pedido...

    responseBuilder.add("O valor total do pedido é: $totalPedido")

    return responseBuilder.endConversation().build()
}
```

# SimpleResponse

```
val responseBuilder = getResponseBuilder(request)

responseBuilder.add(
    SimpleResponse()
        .setDisplayText("Texto Exibido na tela")
        .setTextToSpeech("Texto Falado")
)

responseBuilder.add("isso é um exemplo de card")

return responseBuilder.build()
```

# Suggestions

```
val responseBuilder = getResponseBuilder(request)
|
return if (!request.hasCapability(Capability.SCREEN_OUTPUT.value)) {
    responseBuilder
        .add("Desculpe, seu dispositivo não suporta essa funcionalidade.")
        .build()
} else {
    responseBuilder
        .addSuggestions(arrayOf("Suggestion chips", "suggestion 1", "suggestion 3"))
        .add(
            LinkOutSuggestion()
                .setDestinationName("Suggestion link")
                .setUrl("https://assistant.google.com")
        ).build()
}
```



# BasicCard

```
val responseBuilder = getResponseBuilder(request)

if (!request.hasCapability(Capability.SCREEN_OUTPUT.value)) {
    return responseBuilder
        .add("Desculpe, seu dispositivo suporta essa funcionalidade.")
        .build()
}

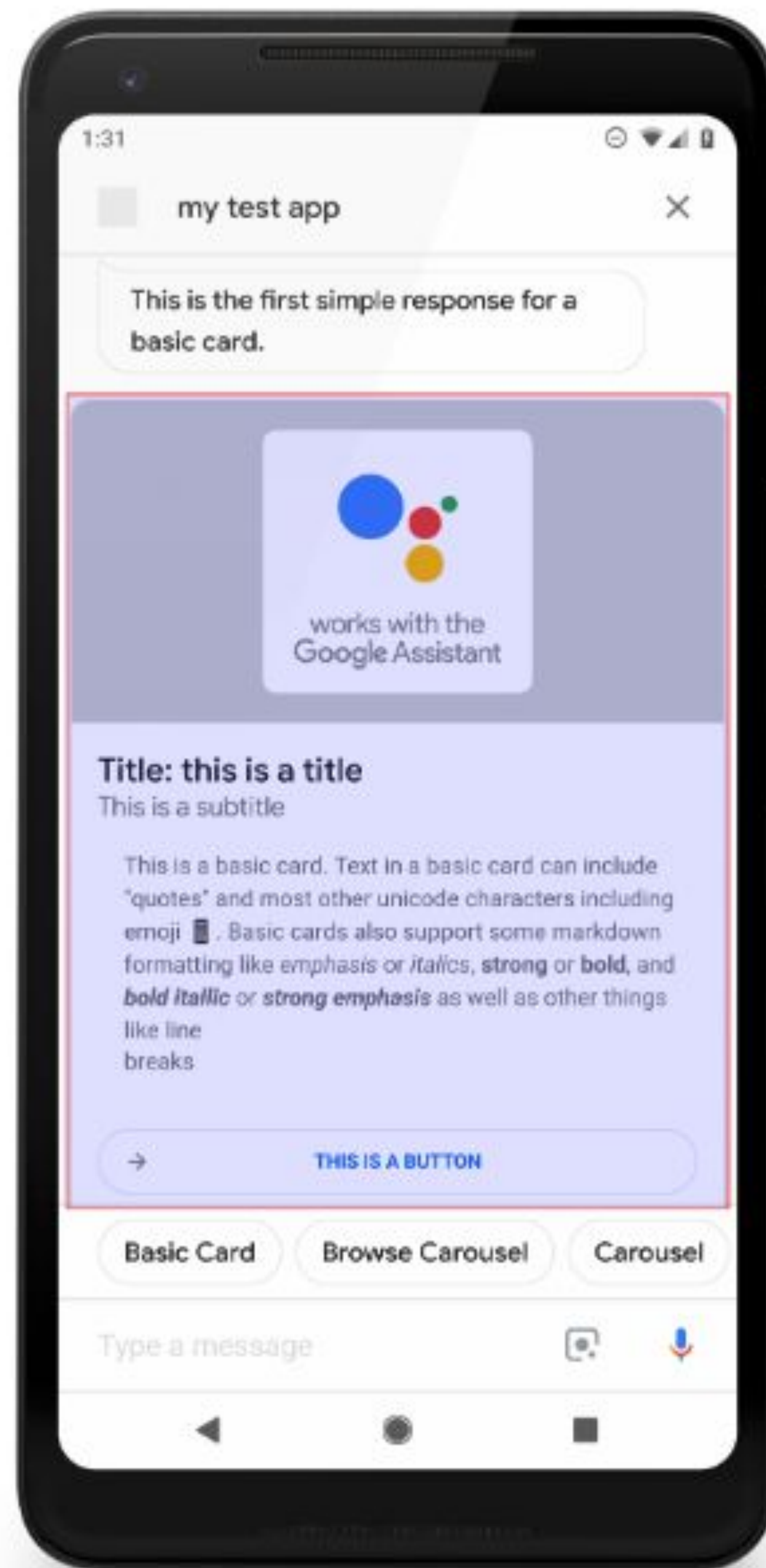
val mButton = Button()
    .setTitle("Texto do botão")
    .setOpenUrlAction(OpenUrlAction().setUrl("https://assistant.google.com"))

val buttons = ArrayList<Button>()
buttons.add(mButton)

responseBuilder
    .add("isso é um exemplo de card")
    .add(
        BasicCard()
            .setTitle("Titulo")
            .setSubtitle("Subtitulo")
            .setFormattedText("Este é um texto longo...")
            .setImage(
                Image()
                    .setUrl("http://example.com/image.png")
                    .setAccessibilityText("Imagem Texto alternativo")
            )
            .setImageDisplayOptions("CROPPED")
            .setButtons(buttons)
    )

return responseBuilder.build()
```





# Responses

Simple response

Rich responses

Basic card

Browsing carousel

Suggestion chips

Media responses

Table cards

Visual selection  
responses

List

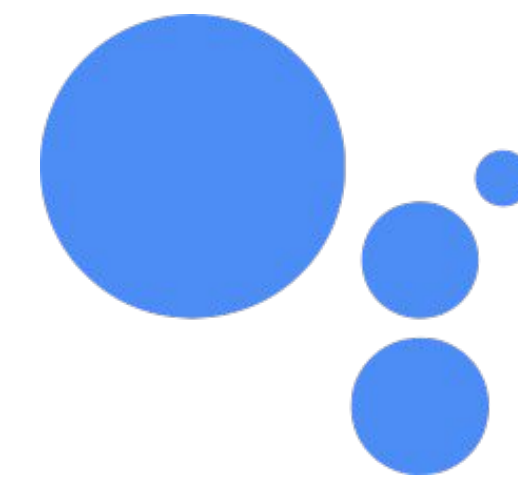
Carousel

Customizing your  
responses

# Desing de Conversas

- Conversação é Inerentemente multimodal
- Ensinar os computadores a serem fluentes em conversas
- Fluxo da conversa
- Contexto
- Personas

# Desing de Conversas



Seis princípios da conversação humana prontos para revolucionar o design da VUI  
por James Giangola Conversation & Persona Design Lead @Google

- *Dê a sua VUI uma personalidade*
- *Mova a conversa para frente*
- *Seja breve, seja relevante*
- Contexto de alavancagem
- *Direcione o foco do usuário através da ordem das palavras e do estresse*
- *Não ensine "comandos" - falar é intuitivo*

“A voz tem sido essencial para a interação humana desde bem antes da história. Mas o que é velho agora é novo: a voz está se tornando essencial para a forma como interagimos com os computadores”

**MARK STEPHEN MEADOWS, AUTOR, ARTISTA E PRESIDENTE DO BOTANIC.IO**

*Fonte: Designing Voice User Interfaces*

# Links / Referências



## Meus Artigos:

Actions on Google: Desenvolvendo Actions para o Google Assistant do zero: [aqui](#)

Actions on Google: usando o Google Assistant a seu favor: [aqui](#)

## Google:

Lib Kotlin/Java: <https://github.com/actions-on-google/actions-on-google-java>

Actions on Google : <https://developers.google.com/actions>

Actions on Google SDK: <https://developers.google.com/assistant/sdk>

Guia Desing Conversas: <https://designguidelines.withgoogle.com/conversation>

Dialogflow: <https://dialogflow.com>

AIY Projects: <https://aiyprojects.withgoogle.com>

Codelabs Assistant : <https://codelabs.developers.google.com/?cat=Assistant>

You can download the fonts at <https://www.fontsquirrel.com/fonts/raleway>



**VAGAS 100+**



<http://bit.ly/zup-tdc-sp>



# Obrigado!



Perguntas?

Você pode me encontrar no

 [@WagnerMessiasC](https://twitter.com/WagnerMessiasC)

