Monitoramento contínuo - Problema 3

Messias J. L. da Silva[1]

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

mjls.junior.js@gmail.com

1. Introdução

Devido a pandemia do coronavírus, houve um grande aumento nos usuários de assistentes virtuais no Brasil. Pensando nisso, a Integrated IP LLC solicitou o desenvolvimento de um assistente, capaz de controlar o protótipo que foi desenvolvido anteriormente, juntamento com a aplicação web. Este novo sistema de automação residencial deve ser capaz de controlar os dispositivos conectados, realizar configurações no controlador dos dispositivos e ainda atualizar o usuário sobre o estado do ambiente.

O objetivo principal deste projeto é implantar uma configuração para monitoramento de variáveis via assistentes virtuais. E como resultado da construção deste projeto é esperado obter um maior entendimento sobre os conceitos do protocolo MQTT, entender os recursos básicos de serviços de assistentes de voz e projetar, desenvolver e colocar em produção um sistema computacional com interface por voz para acesso a um dispositivo.

Dentre os principais requisitos solicitados estão: o uso do protocolo MQTT, armazenamento do histórico de eventos por 24h, verificação da conexão com a nuvem e do estado do dispositivo, configuração do tempo de notificação e emissão da notificação tanto na aplicação web como no assistente, além das operações de controle dos dispositivos.

Este trabalho está dividido em 5 seções, sendo esta a Introdução e as demais são a Fundamentação Teórica, Metodologia, Resultados e Discussões e Conclusão, onde todo o processo de desenvolvimento da solução para o problema proposto será detalhado.

2. Fundamentação Teórica

2.1. Assistente virtual

Um assistente virtual inteligente é um software que responde a comandos de voz, como por exemplo, a Siri, da Apple, a Alexa, da Amazon, o OK Google, do Google, dentre outros. As funções de um assistente virtual agem como facilitador de questões cotidianas na vida de uma pessoa

2.2. Monitoramento contínuo

O monitoramento contínuo trata de um conjunto de dispositivos que promovem o acesso a dados em tempo real. Isso permite a verificação das métricas pré-determinadas para cada equipamento, ambiente, ou item que precisa ser observado.O monitoramento contínuo é um dos itens fundamentais para antecipar paradas ou problemas. Uma vez que permite identificar anomalias ou falhas nos equipamentos.

2.3. Google Actions

Actions on Google é uma plataforma de desenvolvimento para o Google Assistant . Ele permite o desenvolvimento de ações por terceiros - aplicativos para o Google Assistant que fornecem funcionalidade estendida.

2.3.1. Ações de conversação

Ações de conversação estendem a funcionalidade do Google Assistant, permitindo que o desenvolvedor crie experiências personalizadas, ou conversas, para usuários do Google Assistant. Em uma conversa, sua Ação de conversação lida com solicitações do Assistente e retorna respostas com componentes de áudio e visuais. Ações conversacionais também podem se comunicar com serviços da web externos com webhooks para adicionar lógica conversacional ou de negócios antes de retornar uma resposta. Uma conversa define como os usuários interagem com uma ação depois que ela é chamada. Você constrói conversas com intenções , tipos , cenas e prompts

2.3.2. Invocation

Uma invocação define como os usuários dizem ao Assistente que desejam iniciar uma conversa com uma de suas Ações. A invocação de uma ação é definida por um único intent que é correspondido quando os usuários solicitam a ação.

2.3.3. Intents

Os intents representam uma tarefa que o Assistente precisa da sua Ação para realizar, como alguma entrada do usuário que precisa de processamento ou um evento do sistema que você precisa controlar. Você usa intenções para ajudar a construir seus modelos de invocação e conversação . Quando esses eventos ocorrem, o tempo de execução do Assistente o associa ao intent correspondente e envia o intent para que sua Ação seja processada.

Ao construir ações, você cria intents do usuário que contêm frases de treinamento, o que amplia a capacidade do Assistant de entender ainda mais. Quando isso ocorre, o Assistant intermedia a comunicação entre o usuário e suas Ações, mapeando a entrada do usuário para uma intenção que possui um modelo de idioma correspondente. O Assistente então notifica suas ações sobre a intenção correspondente, para que você possa processá-la em uma cena .

Ao construir intents do usuário, você especifica os seguintes elementos:

- Uma designação de intenção global: as intents que marcadas como globais estão qualificados para invocação de link direto, ou seja, o assistente e a operção são invocadas numa mesma frase.
- As frases de treinamento são exemplos do que um usuário pode dizer para corresponder à intenção. Fornecer um grande conjunto de exemplos de alta qualidade aumenta a qualidade da intenção e a precisão da correspondência.

- Os parâmetros são dados que você deseja extrair da entrada do usuário. Para criar um parâmetro, você anota frases de treinamento com tipos para notificar o mecanismo NLU de que deseja que partes da entrada do usuário sejam extraídas. Você pode usar tipos de sistema ou criar seus próprios tipos personalizados de parâmetros.

2.3.4. Scenes

Em combinação com as intenções , as cenas são o outro bloco de construção principal do seu modelo de conversação . As cenas representam estados individuais de sua conversa e seu objetivo principal é organizar sua conversa em blocos lógicos, executar tarefas e retornar prompts aos usuários.

2.3.5. Prompts

Os prompts definem como sua ação processa as respostas aos usuários e como sua ação solicita que eles continuem. Conforme você constrói sua ação, você pode adicionar prompts a invocações e a vários lugares dentro das cenas.

2.3.6. Webhook

Para dar a você ainda mais flexibilidade na construção de ações, você pode delegar lógica para serviços da web HTTPS (atendimento). Suas ações podem acionar webhooks que fazem solicitações a um endpoint HTTPS. Alguns exemplos do que você pode fazer no atendimento incluem:

- Gerar um prompt dinâmico com base nas informações fornecidas pelo usuário;
- Colocar um pedido em um sistema externo e confirmar o sucesso;
- Validando slots com dados de back-end.

As ações podem acionar um webhook em intents ou cenas de invocação, que envia uma solicitação ao seu endpoint de cumprimento. Seu cumprimento contém manipuladores de webhook que processam a carga JSON na solicitação.

2.4. Cloud functions

O Google Cloud Functions é a solução de computação sem servidor do Google para criar aplicativos com base em eventos. É um produto conjunto entre as equipes do Google Cloud Platform e do Firebase.

Para os desenvolvedores do Google Cloud Platform, o Cloud Functions funciona como uma camada de conexão. Com ele, é possível desenvolver a lógica entre os serviços do Google Cloud Platform (GCP) por meio da detecção e da resposta a eventos.

2.5. Node.js

O Node.js pode ser definido como um ambiente de execução Javascript server-side. Isso significa que com o Node.js é possível criar aplicações Javascript para rodar como uma aplicação standalone em uma máquina, não dependendo de um browser para a execução

3. Metodologia

Para iniciar o desenvolvimento desta solução inicialmente foi desenvolvido um diagrama de componentes do sistema. O diagrama está representado na Figura 1, este diagrama é uma atualização do que foi criado no Problema 2, adicionando apenas os componentes que foram utilizados no assistente virtual.

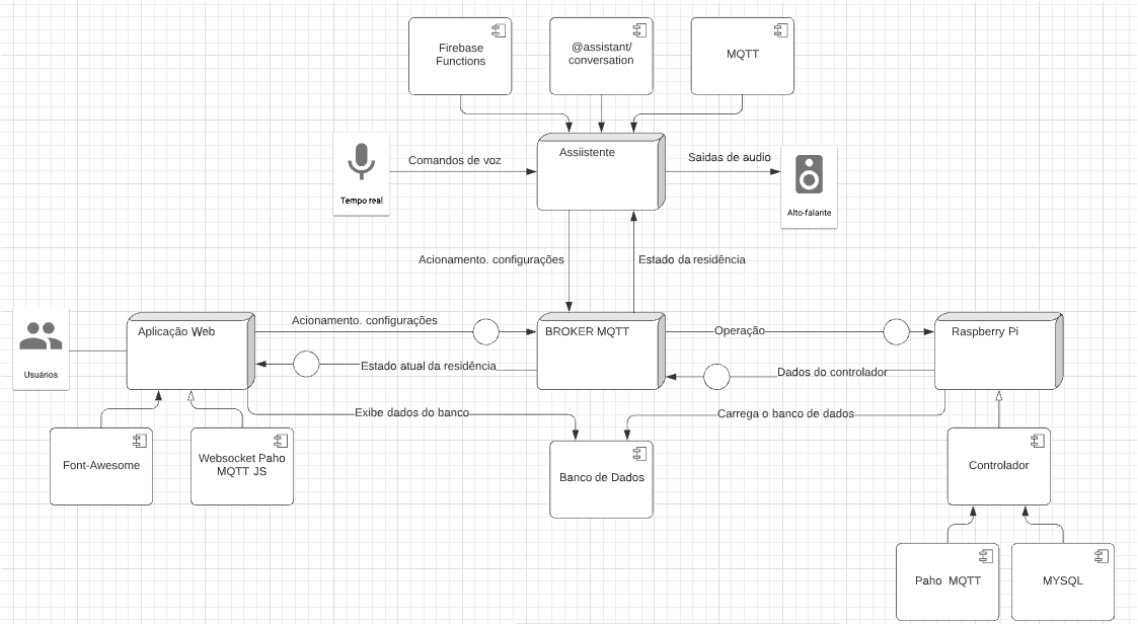

Figura 1. Diagrama de componentes do sistema.

Durante as sessões tutoriais, as principais ideias foram usar a assistente da Amazon, a Alexa, mas devido a facilidade de acesso a documentação, entendimento do funcionamento da ferramenta, foi utilizado o assistente do Google.

Iniciando o desenvolvimento , primeiro é necessário criar um projeto no Google Actions, para isto deve-se ter uma conta no google e estando logado na conta, basta acessar o console do Google Actions e então criar o novo projeto. Ao criar o novo projeto, deve-se escolher o nome, idioma e o país, que foram AURE, Português e Brasil respectivamente, e em seguida escolher a opção de ações customizadas e projeto em branco, para criar o projeto do zero. Feito isso será exibido o painel inicial do Actions console (Figura 2), já na área de desenvolvimento onde é escolhido o nome de invocação para as ações, o nome escolhido nesta solução foi “Controlador”, ou seja, o usuário usa o nome controlador para interagir com o assistente.

Posteriormente partiu-se para a criação das intents, que representa a ação que será realizada. Para a criação de intents basta acessar o menu lateral na opção Custom intents e adicionar nova intent, em seguida será necessário dizer se a intenção é global ou não, (nesse projeto todas as intents são globais para que o usuário possa acionar diretamente uma intenção) e em seguida adicionar algumas as frases de treinamento, que são as frases que acionarão a intent que está sendo criada (Figura 3). E em seguida, caso necessário, pode ser inserido alguns parâmetros para as intents, que são dados informados pelo usuário que serão usados na execução da intenção (Figura 4). por fim, basta salvar as alterações.

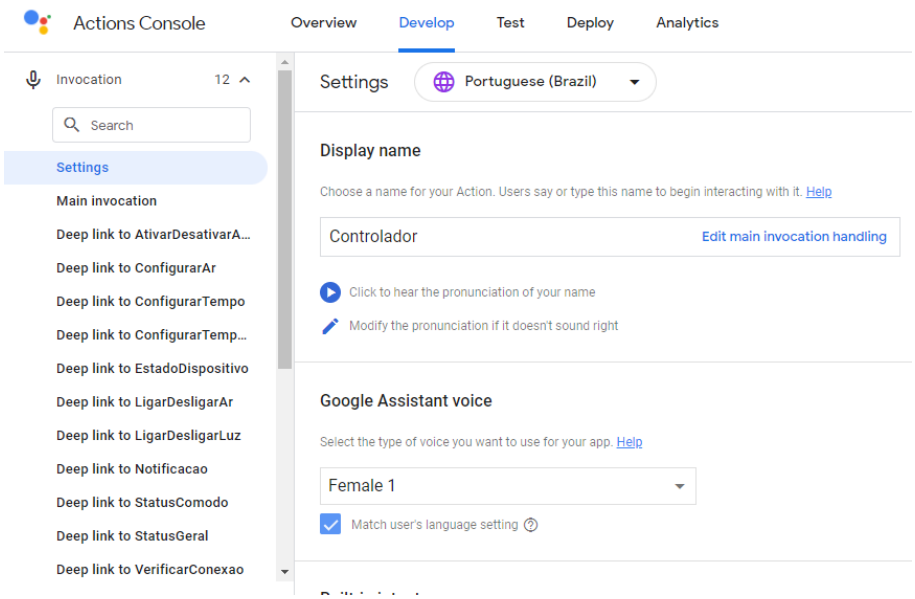

Figura 2. Início do Actions Console, área de configuração do nome da ação .

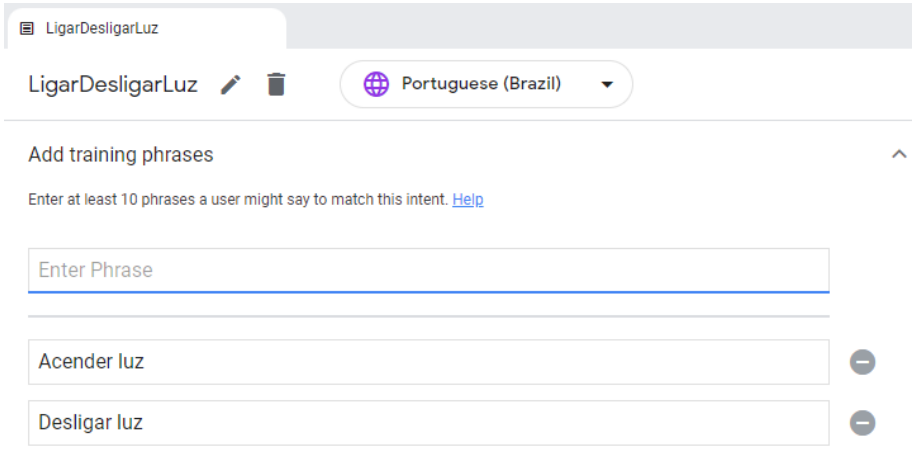

Figura 3. Campo para adicionar frases de treinamento para a intenção .

No Google Action os parâmetros são chamados de Tipos e eles podem ser os tipos padrão do sistema ou podem ser criados novos tipos. Para criar um novo tipo, é necessário acessar o menu lateral na opção Types e adicionar um novo tipo, em seguida podem ser feitas algumas configurações, mas o mais importante é definir quais entradas do usuário representam este tipo, várias entradas podem ser inseridas.

Com as intenções devidamente criadas, é necessário agora associar a intenção a uma cena, que irá realmente executar a tarefa. Para criar uma intenção, basta acessar o menu lateral e criar uma nova cena, ou editar a intenção global e adicionar uma nova transição para a uma cena (que será criada ao adicionar o nome na área da transição) Figura 5.

Com a cena criada, é necessário adicionar algumas configurações, como por exemplo adicionar os slots, que são os parâmetros que serão usadas no cumprimento da intenção, na Figura 6 pode ser observado as opções de configuração e na parte de Slot

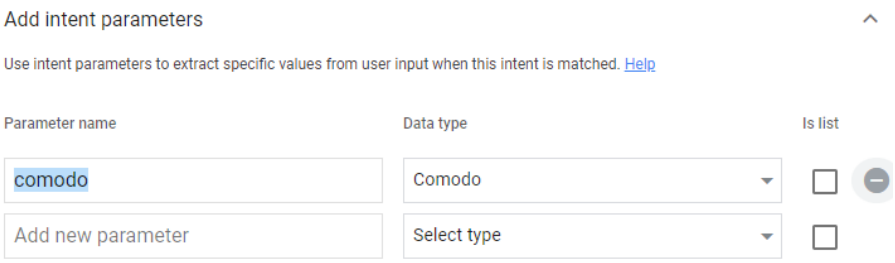

Figura 4. Campo para adicionar parâmetro para a intenção.

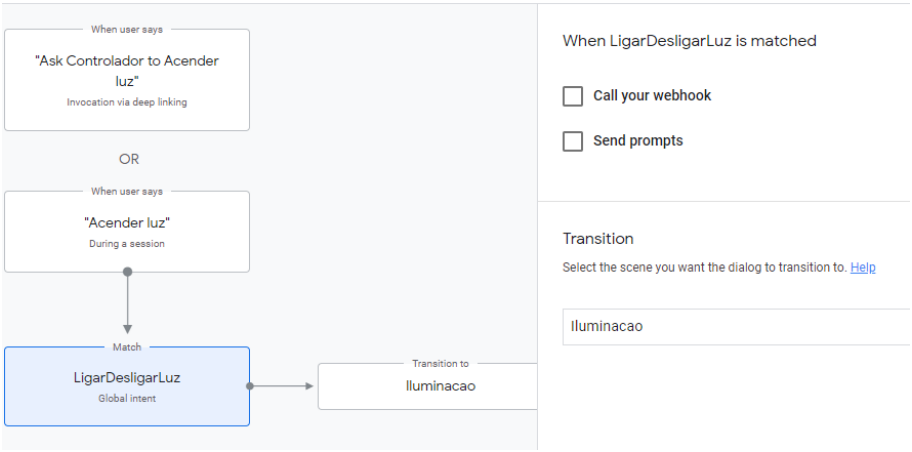

Figura 5. Campo para adicionar parâmetro para a intenção.

filling, basta adicionar uma opção e escolher o tipo que representa o parâmetro, no caso foi o tipo cômodo, pode ser acionado um prompt, que pode perguntar qual é o parâmetro, e também dizer se este parâmetro é obrigatório ou não. Em Conditions, pode-se realizar uma operação caso a condição seja satisfeita, neste caso, se o status da cena for “FINAL”, será chamado o webhook (Figura 7).

Neste projeto foram criadas 11 intents e uma cena para cada intent, e dois tipos foram usados como parâmetros, cada intent representa um requisito do sistema e estão listados na Tabela 1. Cada intenção faz uma chamada ao webhook, que publica as mensagens via MQTT para o Raspbarry pi onde acontece o gerenciamento dos dispositivos.

O Webhook, que é o sistema de cumprimento de intenções disponibilizado pelo Google Actions, está hospedado no Google Cloud Functions e utiliza as funções do Firebase para interagir com o assistente. O código referente ao webhook foi escrito no próprio console do Google Actions, no arquivo Index.js, que utiliza o arquivo Package.json para as informações do webhook e para adicionar as dependências necessárias, que neste caso foram as bibliotecas MQTT e e System-sleep. No webhook, é possível criar respostas dinâmicas para as intenções e também se comunicar com dispositivos externos, que neste caso é feito via MQTT.

O Webhook ao se conectar com o MQTT se inscreve no tópico referente ao sistema de automação residencial que recebe mensagens do Raspberry Pi, o tópico específico para o assistente é: AutomacaoResidencial/Assitente/#. Feito isso, toda mensagem recebida

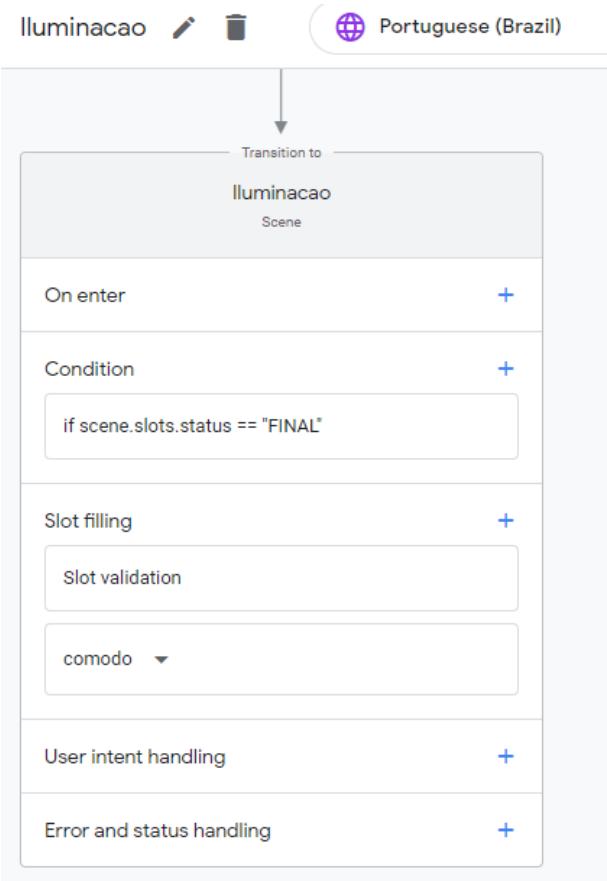

Figura 6. Área de configuração de cena.

neste tópico é adicionada numa lista de mensagens que será tratada para servir como resposta para as solicitações das intents. O tratamento da mensagem ocorre em uma função que foi criada para este propósito (Figura 8), assim que a função é chamada, a mensagem recebida é guardada em uma variável e é removida da lista, já que não será mais usada posteriormente, o tópico é separado da mensagem e em seguida é verificado em qual tópico a mensagem foi recebida, após isso é definido a resposta que depende da mensagem recebida em cada tópico.

Quando o webhook é chamado, um handle é acionado, é feita uma publicação via MQTT para realizar alguma configuração no Raspberry Pi, a Figura 8 mostra um trecho do código do handle que é acionado quando uma intent de Iluminação é acionada. Ao publicar a mensagem, é necessário esperar algum tempo para receber a mensagem, tratá-la e enviar retornar a resposta à conversa, essa espera é feita através da função sleep, que aguarda o tempo enviado por parâmetro. Isto acontece para todas as operações que necessitam de uma resposta do Raspberry Pi, as que não precisam retornam a resposta de acordo com o estado das variáveis do webhook. A variável app representa a aplicação de conversação que chama a função handle para lidar com a chamada, o termo conv representa a conversa, que tem os parâmetros que foram passados e onde as frases podem ser adicionadas para serem emitidas pelo assistente

Para acada ação que pode ser executada, há apenas um tópico, por exemplo, o mesmo tópico é usado tanto para ligar, quanto para desligar uma lâmpada, já que no

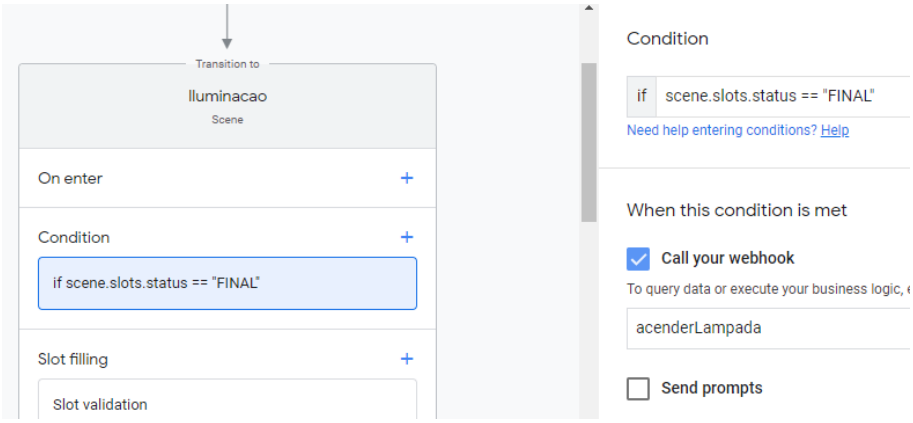

Figura 7. Tratamento de condição para chamar o webhook.

Tabela 1. Relação entre as Intents e as chamadas ao webhook

Intent/Cena	Chamada Webhook
Acender e apagar lâmpadas	acenderLampada
Ligar e desligar ar-condicionado	ligarAr
Ativar e desativar alarme	ativarAlarme
Configurar a faixa de tempo do ambiente interno	configurarTempo
Configurar ar-condicionado	configurarAr
Configurar tempo de notificação	configTempoNotif
Consultar notificações	notificacoes
Consultar o estado do dispositivo	estadoDispositivo
Consultar estado do cômodo	statusComodo
Consultar estado da residência	statusGeral
Consulta conexão	verificarConexao

Raspberry Pi, acontece apenas a troca de estado, então não foi necessário criar tópicos específicos para ligar e desligar um dispositivo, bem como intents diferentes para cada situação.

Partindo para o requisito de conexão com o dispositivo, o Raspberry Pi publica a todo momento no tópico “AutomacaoResidencial/Assistente/Monitoramento” para que as aplicações conectadas a ele, aplicação web e assistente, sabiam o momento em que ele se desconectar, já que não irá receber as publicações de monitoramento. Então, toda vez que a aplicação recebe uma mensagem no tópico especificado, inicia um cronômetro, que tem os segundos incrementado a cada 1000 milissegundos até o valor configurado em minutos, que por padrão é 1 minuto, se o cronômetro atingir o tempo configurado (não chegou nenhuma mensagem no tópico), uma mensagem referente a desconexão do dispositivo, é exibida na aplicação web e adicionada na lista de notificações do assistente.

No Raspberry Pi foi criada uma função chamada monitoramento contínuo que é executada por uma Thread, enquanto o código estiver sendo executado. Esta função publica continuamente uma mensagem no tópico de monitoramento, a mensagem que é enviada para a web contém apenas o tempo de notificação, já para o assistente contém

```
/*Funcao que trata as mesagens recebidas*/
function trataMensagens(){
  var conteudo, topic, message, dados;
    conteudo =  messages.shift();
    dado = conteudo.split("|");
    topic = dado[0];
    message = dado[1];
    //Verifica em qual tópico a mensagem chegou exibe e armazena a informação sobre o estado do dispostivo
    if (topic == "AutomacaoResidencial/Assistente/Lampada/'Jardim'") {
        content = message.split(":");
        if(Number(content[1])==0){
            resposta = "A iluminação do Jardim está desligada!";
        } else if(Number(content[1])==1){
            resposta = "A iluminação do Jardim está ligada!";
        }
    } else if (topic == "AutomacaoResidencial/Assistente/Lampada/'Garagem'") {
```

Figura 8. Campo para adicionar frases de treinamento para a intenção .

```
/*Hanlder chamado quando a cena de iluminacao é invocada*/
app.handle('acenderLampada', conv => {
  const comodo = conv.session.params.comodo; //obtem o comodo
  const topico = estadoLampada + "'"+ comodo +"'"; //concatena o comodo ao tópico
  client.publish(topico,"ASSISTENTE");//publica a mesagem no tópico
  sleep(5000);
  conv.add(resposta);
});
```

Figura 9. Campo para adicionar frases de treinamento para a intenção .

o tempo de notificação e o horário em que a publicação está acontecendo, para que é o horário que será informado na notificação caso o dispositivo seja desconectado.

Para cumprir o requisito de armazenamento de histórico de eventos, foi criada uma função chamada salva_log, que armazena em um arquivo de texto (.txt) todas as ações que foram executadas e horário da ocorrência. Como deve ser armazenado o histórico das últimas 24 horas foi pensando em salvar o arquivo com a data, no caso, fica armazenado um arquivo para cada dia.

4. Resultados e Discussões

Basicamente a solução aqui proposta é um assistente virtual devido com o Google Assistant, que por sua vez interage com o Raspberry Pi, para realizar as principais funcionalidades do sistema. O Raspberry atua como dispositivo central, já que atende tanto às solicitações do assistente, quanto da aplicação web que foi desenvolvida anteriormente.

Para o desenvolvimento desta solução foram utilizados os principais conceitos referente ao assistente do Google tais como o de ações, intenções, cenas, tipos, prompt e webhook, que foram descritos na Fundamentação Teórica. Esses conceitos foram essenciais no desenvolvimento já que eles são a base para a criação de assistentes inteligentes usando o Google.

A Figura 10 descreve muito bem o funcionamento do assistente, onde o usuário aciona o assistente falando “Controlador” ou “Falar com Controlador”, o assistente traduz o áudio em texto, fazendo a compreensão de linguagem natural e envia para o webhook que é o serviço de cumprimento que está no hospedado no Google Cloud Functions. O webhook faz toda a lógica necessária e envia a resposta para o assistente que retorna ao usuário em forma de áudio.

Para o funcionamento do assistente algumas frases específicas foram utilizadas para acionar as intents, algumas destas frases serão listadas a seguir, juntamente com o

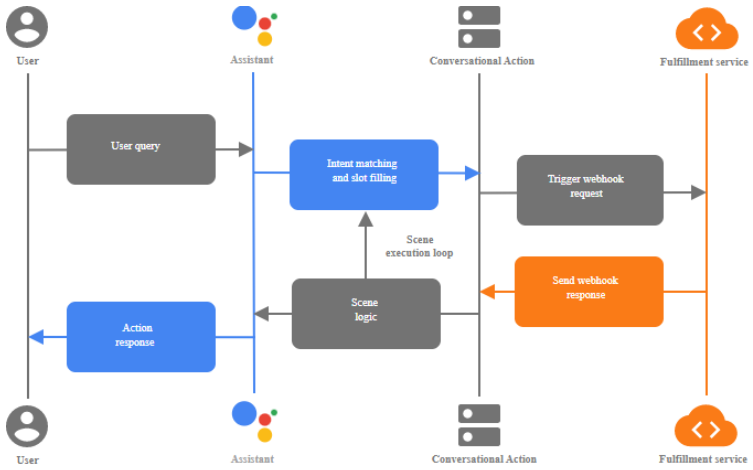

Figura 10. Fluxo de conversa no assistente.

retorno esperado para cada uma delas.

- Iluminação:
  Entrada: ”Falar com controlador para ligar luz da sala”
  Saída: ”A luz da sala está acesa!”

  Entrada: ”Falar com controlador para apagar a luz da sala”
  Saída: ”A luz da sala está apagada!”

- Ar-condicionado:
  Entrada: ”Controlador ligue o ar-condicionado”
  Assistente pergunta: ”Qual cômodo?”
  Usuário responde: ”sala”
  Saída: ”O ar-condicionado da sala está ligado!”

- Alarme:
  Entrada: ”Ativar alarme”
  Saída: ”O alarme está ativado!”

  Entrada: ”Falar com controlador para desligar o alarme”
  Saída: ”O alarme está desativado!”

- Configuração do ar-condicionado:
  Entrada: ”Falar com controlador para configurar o ar-condicionado”
  Assistente pergunta: ”Qual a temperatura máxima?”
  Usuário responde: ”22 graus”
  Assistente pergunta: ”Qual a temperatura mínima?”
  Usuário responde: ”16”
  Assistente pergunta: ”Qual a o tempo de espera?”
  Usuário responde: ”5 minutos”
  Saída: ”Configuração enviada!”

- Configuração do ambiente interno:
  Entrada: ”Configurar o ambiente interno”
  Assistente pergunta: ”Qual o horário inicial?”
  Usuário responde: ”17 horas”
  Assistente pergunta: ”Qual o horário final?”
  Usuário responde: ”22”
  Saída: ”Configuração enviada!”

- Configuração do tempo de notificação:
  Entrada: ”Configurar notificação”
  Assistente pergunta: ”Qual o tempo em minutos?”
  Usuário responde: ”2 minutos”
  Saída: ”Configuração enviada!”

- Consultar estado do dispositivo:
  Entrada: ”Controlador qual o estado do dispositivo?”
  Saída: ”O dispositivo está conectado”

- Consultar notificações:
  Entrada: ”Controlador quais são minhas notificações?”
  Saída: ”O alarme foi acionado as 17 horas e 45 minutos. O dispositivo foi desconectado as 20 horas e 10 minutos”

- Consultar estado do cômodo:
  Entrada: ”Controlador qual o estado da sala?”
  Saída: ”A luz e o ar-condicionado estão ligados!”

- Consultar estado da casa:
  Entrada: ”Controlador qual o estado da casa?”
  Saída: ”Há 3 dispositivos ligados: Alarme. Ar-condicionado da sala. Luz da sala”

- Consultar conexão com a nuvem:
  Entrada: ”Controlador verificar conexão com a nuvem?”
  Saída: ”O assistente está conectado ao M Q T T”

Diversas frase foram inseridas nos intents, mas caso o assistente não reconheça uma delas será solicitado para tentar novamente, as perguntas feitas pelo assistente devem ser respondidas de acordo com o que é perguntado, caso contrário o assistente não irá reconhecer, os cômodos cadastrados são: jardim, garagem, sala, cozinha, quarto 1, 2 e 3 e o banheiro; todos as possibilidade de horas foram cadastradas, sendo que os minutos não foram configurados.

Para o funcionamento do assistente, o código referente ao controlador deve está sendo executado no Raspberry Pi, caso contrário o assistente dirá que algo de errado aconteceu, caso uma ação que depende dele seja acionada.Para compilar e executar o

controlador é preciso utilizar a biblioteca Paho-MQTT para a linguagem C, Thread e MySQL . Execute o comando abaixo para fazer esta compilação:

gcc controlador3.c -o controlador3 -lpaho-mqtt3c -lpthread -lmysqlclient -Wall

Após a compilação, execute utilizando o comando abaixo:

./controlador2

Para verificar o funcionamento do programa, quando os comando por voz foram dados, o resultado foi acompanhando tanto no Raspberry Pi, com a exibição das mensagens que chegaram e das operações que foram efetuadas, como na aplicação web, onde o ícones referentes aos dispositivos mudavam de cor, à medida que o seu estado era alterado no Raspberry.

Figura 11. Campo para adicionar frases de treinamento para a intenção .

5. Conclusão

O projeto aqui apresentado é capaz de controlar o sistema de automação residencial, cumpre o objetivo proposto e todos os requisitos que foram solicitados, havendo apenas uma inconsistência em relação ao histórico de eventos que deveria ser salvo apenas com as operações referentes às últimas 24h e que neste projeto o histórico é mantido para cada dia. Além dos requisitos propostos, é possível também emitir uma notificação caso haja um alerta de alarme, verificar o estado de toda a residência, não apenas de um dos cômodos e também manter os estados do dispositivo ao recarregar a página web.

Para trabalhos futuros, seria interessante desenvolver uma aplicativo, assim como há a aplicação Web, para dar mais acessibilidade e mobilidade aos usuários, e também acrescentar novas funcionalidades como controle de estoque da geladeira e verificar o consumo de energia.

Referências

Google. Ações de conversação: Google Assistente, 11 ago. 2020. Disponível em: https://developers.google.com/assistant/conversational/overview. Acesso em: 28 nov. 2021.

Induca. A MANUTENÇÃO PREDITIVA E O MONITORAMENTO CONTÍNUO, 10 out. 2015. Disponível em: https://sensorweb.com.br/manutencao-preditiva-e-monitoramento-continuo/. Acesso em: 28 nov. 2021.

SILVA, Douglas da. A. Assistente virtual inteligente: o que é como usá-lo nos negócios, 31 maio 2021. Disponível em: https://www.zendesk.com.br/blog/assistente-virtual-inteligente-o-que-e/. Acesso em: 28 nov. 2021.

Lenon. Node.js – O que é, como funciona e quais as vantagens, 5 set. 2018. Disponível em: https://www.opus-software.com.br/node-js/. Acesso em: 28 nov. 2021.