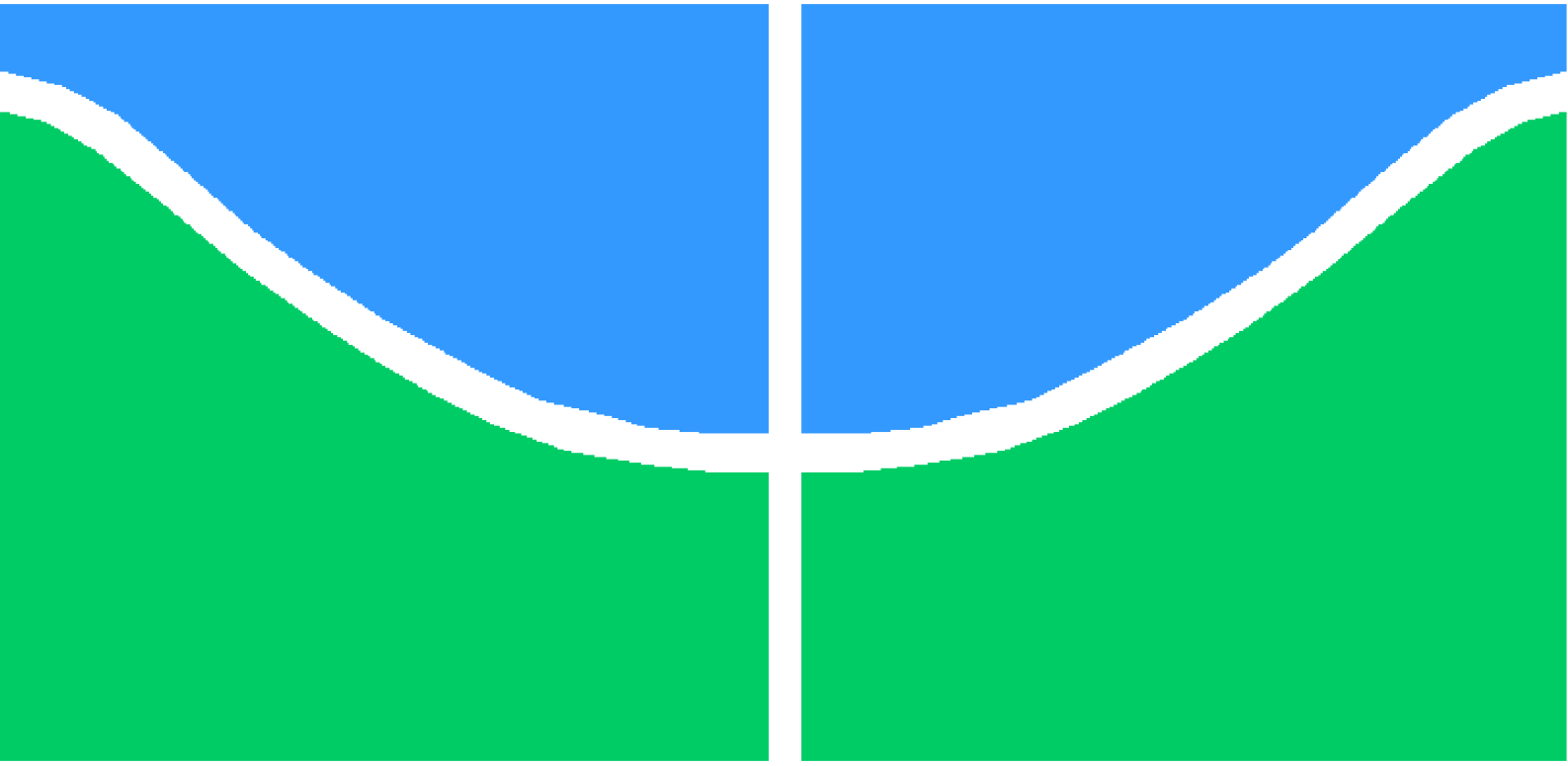

Universidade de Brasília – UnB
Faculdade UnB Gama – FGA
Engenharia de Software

# Automação Residencial com foco em Cenários

Autor: Artur Bersan de Faria
Orientador: Dr. Renato Coral Sampaio

Brasília, DF
2018

Artur Bersan de Faria

# Automação Residencial com foco em Cenários

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília – UnB

Faculdade UnB Gama – FGA

Orientador: Dr. Renato Coral Sampaio

Brasília, DF

2018

Artur Bersan de Faria
Automação Residencial com foco em Cenários/ Artur Bersan de Faria. – Brasília, DF, 2018-
87 p. : il. (algumas color.) ; 30 cm.

Orientador: Dr. Renato Coral Sampaio

Trabalho de Conclusão de Curso – Universidade de Brasília – UnB
Faculdade UnB Gama – FGA , 2018.

1. Automação Residencial. 2. Cenários. I. Dr. Renato Coral Sampaio. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Automação Residencial com foco em Cenários

CDU 02:141:005.6

Artur Bersan de Faria

# Automação Residencial com foco em Cenários

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 14 de dezembro de 2018:

---

**Dr. Renato Coral Sampaio**
Orientador

---

**Dr. Fernando William Cruz**
Convidado

---

**Dr. Tiago Alves da Fonseca**
Convidado

Brasília, DF
2018

*Este trabalho é dedicado às pessoas que acreditam que com esforço, estudo e persistencia, é possível construir tecnologias que fazem desse mundo um lugar melhor.*

# Agradecimentos

Agradeço esse trabalho primeiramente a minha família que me deu base para a minha caminhada até aqui me oferecendo uma boa educação e apoio a tudo que precisei para chegar até aqui. Agradeço ao laboratório LAPPIS que me proporcionou minha primeira experiência na área, gerando conhecimento para que eu pudesse pegar projetos maiores, sempre me dando espaço para meu crescimento profissional sendo flexível quando era necessário gastar um tempo para alguma atividade na faculdade. Agradeço aos meus companheiros de curso que sempre me ajudaram nessa caminhada e proporcionaram diversos momentos de alegria e descontração e muito conhecimento agregado. Agradeço aos meus professores que deram "o caminho das pedras", muitas vezes colocando mais "pedras no caminho", mas que me motivaram a buscar cada vez mais por mais conhecimento, principalmente aos professores Paulo Meireles, Carla Rocha, Fábio Mendes, Sérgio Freitas, Milene Serrano, Fernando Cruz, Tiago Alves e Renato Coral. E quero fazer um agradecimento especial a minha namorada, Iasmin Mendes, que me apoiou muito no decorrer desse trabalho, me ajudando a revisá-lo, procurar referências, estruturar frases, controlar meus prazos e a me motivar a fazer um trabalho melhor.

*"A IoT irá impulsionar a economia tornando a vida de nossos cidadães melhor"*
*Mário Campolargo*

# Resumo

O presente trabalho trata-se da integração de multiplos serviços de IoT na construção de cenários em diversos ambientes, desde residencias a locais de trabalho, visando automatizar aspectos de ilumininação juntamente com a seleção personalizada de músicas, além de construir uma arquitetura que possibilitar a inserção de um serviço em uma solução de cenários completos. Para tal, adaptou-se o *hub open source* Home Assistant de forma que este controla múltiplos serviços que geram o cenário selecionado pelo usuário, utilizando a camada de Home Automation para este desenvolvimento.

**Palavras-chaves**: automação, cenários, Home Assistant, *hub*, Internet das Coisas, IoT.

# Abstract

The present work deals with the integration of multiple IoT services in the construction of scenarios in different environments, from residences to workplaces, aiming to automate aspects of lighting with personalized selection of music, as well as to build an architecture that allows the insertion of service in a complete scenario solution. To do this, the Home Assistant hub has been adapted so that it controls various services that generate the user-selected scenario by using the Home Automation layer for that development.

**key-words**: automation, scenarios, Home Assistant, hub, Internet of Things, IoT.

# Lista de ilustrações

# Lista de quadros

# Lista de abreviaturas e siglas

| | |
|---|---|
| AMQP | Advanced Message Queuing Protocol |
| API | Application Programming Interface |
| HTTP | Hypertext Transfer Protocol |
| IBM | International Business Machines |
| IFTTT | If this, then that |
| IoT | Internet of Things |
| MOM | Message Oriented Middleware |
| MQTT | Message Queuing Telemetry Transport |
| REST | Representational State Transfer |
| SOA | Service Oriented Architecture |
| SOAP | Simple Object Access Protocol |
| USB | Universal Serial Bus |
| SD | Secure Digital |
| IP | Internet Protocol |
| IPv4 | Internet Protocol versão 4 |
| WiFi | Wireless Fidelity |
| SSH | Secure Shell |
| HDMI | High-Definition Multimedia Interface |
| PWA | Progressive Web Apps |
| IOS | Iphone Operating System |

# Sumário

# 1 Introdução

A sociedade atual encontra-se em uma fase de transição para um novo modelo social denominado Sociedade Digital, no qual o ser humano passa a habitar não somente um meio ambiente, mas um meio ambiente digital (CATARINA, 2018), na qual a IoT se torna um conceito base e fundamental para a construção dessa nova sociedade. Assim, objetos comuns do dia-a-dia, como a televisão e a geladeira, passam a possuir acesso a recursos como internet e diversos sensores, se tornando capazes de prover e consumir informação, além de permitir que os próprios dispositivos tomem decisões perante a determinada situação na qual é submetido.

Nesse contexto, surgem os ambientes inteligentes caracterizados por permitir a interação do usuário com os dispositivos espalhados por todo o local no qual ele se encontra. Propiciando além da interação usuário-dispositivo, a interação entre os próprios dispositivos e abrindo espaço para a personalização das suas ações de acordo com comportamentos previsíveis (FREITAS et al., 2012). Assim, surge a possibilidade do usuário customizar esses dispositivos não unicamente com suas preferências, mas aplicando técnicas voltadas a melhoria de sua qualidade de vida e produtividade.

Por meio de pesquisas experimentais, nota-se que a música e as cores têm se mostrado fatores que influenciam o ser humano. Podendo as mesmas, por meio de técnicas como a cromoterapia[1] e a musicoterapia[2] melhorar a qualidade de vida e a produtividade das pessoas (ROCHA; BOGGIO, 2013; CARVALHO, 2013). Dessa maneira, enxerga-se nos ambientes inteligentes, uma forma para a construção de cenários, voltados a utilizar recursos como os citados, para estimular as áreas cognitivas, psíquicas e físicas do ser humano.

No mercado atual existem dois ramos de desenvolvimento para pessoas que buscam explorar a criação desses cenários. O primeiro consiste em produtos de código proprietário como a Alexa[3] e o Google Home[4]. Estes *hubs*[5] fornecem a integração entre vários aparelhos - lâmpadas, alarmes, mídias, etc. - permitindo a construção de cenários bem elaborados. Juntamente com esses *hubs* é oferecido uma API, a qual permite outros desenvolvedores

[1] Técnica que visa auxiliar o tratamento de doenças usando das cores como fator estimulante.
[2] Técnica que utiliza a música na prevenção e melhoria de problemas relacionados a saúde mental.
[3] Assistente inteligente fornecido pela Amazon para controle por voz, integração com aplicativos, previsão do tempo e outras funcionalidades que visam facilitar a vida do usuário. Mais informações em: <https://developer.amazon.com/alexa>
[4] Assistente digital da Google Inc. com controle por voz, integração com aplicativos e outras funcionalidades. Mais informações em: <https://play.google.com/store/apps/details?id=com.google.android.apps.chromecast.app>
[5] Um hardware ou software que conecta dispositivos em uma rede de automação residencial e controla as comunicações entre eles.

explorarem a construção dos cenários consumindo desta. Contudo, essa abordagem limita certos aspectos do desenvolvimento na área, sendo o desenvolvedor impossibilitado de adequar o código de automação às necessidades do seu projeto solução. Além do mais, o desenvolvedor encontra-se restrito a utilizar o *hardware* e os dispositivos de IoT que a empresa selecionada disponibiliza para a solução.

O segundo ramo consiste em empresas que possuem projetos com código *open source* as quais expandem o espaço para o desenvolvimento de propostas em cima de automação residencial. Entre elas estão alguns nomes como Domoticz[6], Calaos[7] e Home Assistant[8]. Nessa abordagem o desenvolvedor tem total flexibilidade sobre o código e *hardware* para adequá-los aos requisitos do projeto que visa construir. Em contrapartida, a confecção de cenários nessas ferramentas ainda é simplista quando comparada às empresas de código privado, sendo possível configurar pequenos detalhes como a luz, ou o som, mas sem uma customização bem elaborada e feita de forma relativamente fácil como acontece nas empresas de código privado.

Tendo em vista a perspectiva de mercado, a proposta deste trabalho é apresentar uma solução que automatize a configuração de cenários utilizando um *hub open source*, permitindo ao usuário final fazer a seleção de múltiplos serviços conectados à internet, para assim, criar o cenário desejado.

## Justificativa

Este trabalho justifica-se pela deficiência no mercado de *hub open source* para criação de cenários que utilizam múltiplos serviços, tais como a seleção de uma *playlist* e um cenário de iluminação ao mesmo tempo.

## Objetivos

### Objetivo Geral

O presente trabalho objetiva contribuir com um *hub open source* de automação residencial, implementando uma base para a construção de cenários, com múltiplos dispositivos a qual contará com configurações de iluminação artificial e de *playlists* adequadas ao ambiente.

[6] Domoticz - consulte: <https://www.domoticz.com/>
[7] Calaos - consulte: <https://www.calaos.fr/en/>
[8] Home Assistant - consulte: <https://www.home-assistan.io/>

### Objetivos específicos

- Definir um *hub* de automação residencial *open source* que melhor se adéque a abordagem, ou seja, centralize o controle dos dispositivos de IoT em um único aplicativo e que facilite a construção de cenários;
- Definir uma lampada *smart* para configurar o ambiente juntamente com o *software* escolhido;
- Definir um *software* para selecionar a *playlist*;
- Definir uma *media play*[9] base para integrar a solução;
- Tornar possível a configuração de intensidade da iluminação usando uma lampada *smart*;
- Tornar possível a configuração da cor da iluminação usando uma lampada *smart*;
- Tornar possível a escolha da *playlist*.

## Organização do Trabalho

Este trabalho foi organizado em 5 capítulos. Iniciando pela fundamentação teórica, na qual são explicados conceitos relevantes para a solução apresentada, seguido da metodologia adotada, uma análise das ferramentas necessárias para a sua execução e por fim os resultados alcançados.

[9] Dispositivo que possibilita o controle e a reprodução de conteúdo de áudio/vídeo.

# 2 Fundamentação Teórica

Esse capítulo irá tratar da teoria relacionada ao trabalho apresentado.

## 2.1 Cromoterapia e Musicoterapia

As cores vem sido estudadas em diversas áreas, como na arquitetura, marketing e no design. Elas trazem benefícios ao usuário, quando a sua escolha é feita corretamente, podendo melhorar cerca de 40% a leitura, entre 55% a 68% o aprendizado e 73% a compreensão (CARVALHO, 2013). Na área da medicina alternativa, o estudo voltado a auxiliar o tratamento de doenças por meio do uso das cores, é conhecido como cromoterapia, e assim como a medicina tradicional, esta busca a saúde física e mental. Essa técnica foi utilizada desde tempos antigos por grandes civilizações como a Pérsia, o Egito e a China (VALNET, 2015).

> O comportamento humano é conduzido por uma resposta à percepção do ambiente através dos estímulos provocados pelo mesmo.
> (FONSECA; RHEINGANTZ, 2009)

O ser humano é influenciado por três aspectos fundamentais: físico, cognitivo e psíquico. A associação adequada desses fatores permite projetar ambientes seguros, confortáveis e eficientes (KOTH, 2013). A integração da iluminação natural e a iluminação artificial pode ser utilizada para trazer boas sensações a pacientes, minimizando dores, sendo um instrumento importante de apoio à saúde (BIANCHI; DAVID; SUETA, 2017), e em contexto geral pode produzir no indivíduo o estado de ânimo que responde a seu desejo ou ação (MANAIA, 2013).

Em contrapartida à cromoterapia, a musicoterapia utiliza a música para trazer benefícios aos usuários podendo melhorar a percepção auditiva, movimento e memória, além de fazer uma correlação entre o estudo da música e linguagem e as emoções que ela pode evocar. Muitos estudos têm apontado para o papel da música como ferramenta de intervenção em diferentes alterações neurológicas como afasia, autismo e dislexia (ROCHA; BOGGIO, 2013).

Sobre a percepção auditiva, quando uma pessoa escuta uma música, são trabalhadas diversas áreas do sistema nervoso, tais como o córtex pré-frontal, córtex pré-motor, córtex motor, córtex somato sensorial, lobos temporais, córtex parietal, córtex occipital, cerebelo e áreas do sistema límbico, incluindo a amígdala e o tálamo (OVERY; SZAKAC, 2009), exigindo o reconhecimento de seus parâmetros básicos (altura, duração, timbre e

intensidade) e as relações entre eles (ROCHA; BOGGIO, 2013). O não reconhecimento dos parâmetro básicos pode ajudar o identificar lesões na porção anterolateral direita do giro de Heschl, lesões no córtex temporal direito (ROCHA; BOGGIO, 2013).

A percepção rítmica de uma música envolve áreas motoras do cérebro, tanto para quem executa a música, quanto para quem escuta, e por esse motivo, trás o engajamento de tarefas motoras, possuindo bons resultados até com pacientes com doença de Parkinson, trazendo a eles um andar mais fluente ou até mesmo permite que eles consigam dançar quando escutam uma música (ROCHA; BOGGIO, 2013).

A música é aplicada em diversos ambientes e contextos, como em filmes, jogos, lojas, bares e diversos outros ramos profissionais. Ela é utilizada com o objetivo de transmitir emoções a quem escuta. Há diversos aplicativos de música como Spotify[1] e Google Music[2] que permitem criar *playlists* de músicas que podem transmitir uma certa emoção para quem escuta, e até mesmo empresas como a Jamendo[3], que oferece uma rádio de música específica para *shopping center*.

Há diversos autores que dizem que a música pode melhorar o desempenho no ambiente de trabalho e estudo. Abbott (2011), Parncutt e e Mcpherson (2002) dizem que a música deve ser utilizada no processo de aprendizagem, pois ela influencia diretamente na absorção de conhecimento por parte do aluno, além de influenciar o meio de comunicação, e a expressão cultural. Standley (2008) afirma que a música ajuda no processo de alfabetização e Rong-Hwa (2011) diz que a musica pode trazer benefícios ao ambiente de trabalho, pois tem grande influência na concentração humana.

Tendo em vista a cromoterapia e a musicoterapia, como a tecnologia pode ajudar a criação de ambientes que combinam música e iluminação que se adequem a necessidade das pessoas? Esse trabalho traz uma abordagem onde ambientes possam ser configurados de acordo com a necessidade das pessoas por meio de cenários, não sendo um objetivo dele apresentar técnicas de cromoterapia e/ou musicoterapia para melhorar esses ambientes, mas sim, tornar possível o estudo dessas técnicas em cima de ambientes inteligentes.

## 2.2 IoT - Internet das Coisas

A IoT - Internet das Coisas se caracteriza pelo uso da inteligência em dispositivos e sistemas conectados, aproveitando os dados coletados a partir de sensores. Espera-se que esta seja a nova revolução da internet, e que convirja para uma nova dimensão de serviços e qualidade de vida (ASSOCIATION, 2014).

Esta mudança mostra-se tão expansiva que até o dinheiro está se tornando digital,

[1] Spotify - consulte: <https://www.spotify.com/br/>
[2] Google Music - consulte: <https://play.google.com/music/>
[3] Jamendo - consulte: <https://www.jamendo.com/>

com o uso de cartões que creditam em tempo real uma operação bancária, ou um aplicativo web e/ou mobile que faz pagamento ou uma transferência bancária. No Quênia, há um aplicativo de *smartphone* chamado M-PESA[4], que realiza pagamentos e armazenamento de valor. Ele possui 9 milhões de usuários, cerca de 40% da população adulta e 23% da população total do país (MAS; RADCLIFFE, 2010). Uma aplicação com este escalão conecta 9 milhões de usuário na internet, manuseando 320 milhões de transferências *Peer to Peer*[5] e para processar toda essa massa de dados é necessário um grande poder computacional (MAS; RADCLIFFE, 2010). Aplicativos como esse; só são possíveis pois o hardware (*smartphone*) tem a capacidade de se conectar a internet.

Na automação residencial, a IoT mostra-se como um conceito que irá tornar a vida das pessoas mais fácil, fornecendo um uso inteligente dos recursos (VERMESAN; FRIESS, 2013). Atualmente já existem diversas soluções no mercado, que exploram a automação por meio de uma central, denominada *hub*, a qual é responsável por conectar os vários dispositivos e gerenciá-los. Mas, seria vantajoso investir em uma central para ter controle de cenários? Se a necessidade de configurar algum cenário se limita ao controle da iluminação, pode ser que não, pois nesse caso existe lâmpadas inteligentes como a LIFX e Hue Philips que trazem na sua solução a possibilidade de configuração de cenários utilizando diferentes lâmpadas, não sendo necessário assim, ter um gasto maior com uma central. Mas quando há necessidade de controlar múltiplos equipamentos ou utilizar sensores, começa a ficar interessante ter uma central, pois com essa tecnologia é possível utilizar diversos dispositivos de diferentes marcas que podem ou não interagir entre si, através de um controle centralizado de um *hub*.

## 2.3 Computação Distribuída

O sistema de processamento distribuído ou paralelo interliga vários computadores independentes através de uma rede, apresentado-se para o usuário como um único sistema (TANENBAUM; STEEN, 2006). Esse tipo de abordagem traz diversas vantagens em cima de um computador comum quando se trata do processamento de uma grande massas de dados. Segundo (TANENBAUM; STEEN, 2006), um sistema distribuído tem como objetivo trazer a aplicação:

- Recursos acessíveis: o principal objetivo de um sistema distribuído é tornar fácil o acesso e o compartilhamento de recursos remotos usuários e aplicações;

[4] M-PESA - consulte: <https://www.mpesa.in/>

[5] Formato de rede de computadores em que a principal característica é descentralização das funções convencionais de rede, onde o computador de cada usuário conectado acaba por realizar funções de servidor e de cliente ao mesmo tempo.

- Escalabilidade: ao acrescentar um novo computador na rede, o sistema como um todo ganha capacidade de processamento quase que proporcionalmente a capacidade de processamento do computador acrescentado;
- Tolerância a falhas: um sistema distribuído estará normalmente disponível continuamente, embora talvez algumas partes possam estar temporariamente fora de funcionamento;
- Transparência: para um sistema distribuído é importante que o fato do processamento e os recursos compartilhados entre diversos computadores sejam vistos pelos usuários e aplicações como um único sistema;
- *Openness*(Abertura): um sistema distribuídos oferece serviços de acordo com regras padrão que descrevem a sintaxe e semântica desses serviços. Tais regras são formuladas como protocolo.

No trabalho em questão será feito uma rede de dispositivos interligados a uma central, formando um sistema distribuído no âmbito de IoT. Cada dispositivo irá enviar e receber informações e fazer seu processamento interno para executar sua tarefa de acordo com os dados recebidos. A central terá o papel de receber os dados vindos de todos os dispositivos da rede e informações vindas do usuário para que seja possível fazer o processamento de tomada de decisão.

Esse sistema assemelha-se ao mecanismo mestre-escravo, onde a central será o mestre e os dispositivos os escravos e o usuário será quem requisita o serviço.

Para atender todos os objetivos de um sistema distribuído será explicado a arquitetura MOM e o protocolo MQTT (arquitetura e protocolo utilizados pelo *hub open source*, o Home Assistant, explicado na Seção 2.9.1), Seções 2.5.2 e 2.7 respectivamente.

## 2.4 *Middleware*

A definição sobre o que venha ser um *middleware* é bastante divergente em diversas situações distintas. A definição que será tratada nesse tópico, é a de *middleware* de integração. Segundo o dicionário online livre da computação de Denis Howe (2012), *middleware* é o *software* que faz a mediação entre um programa de um aplicativo e uma rede, já a IBM diz (2015) que o *middleware* é a camada de *software* que interliga um sistema de engajamento (como uma aplicação mobile) a um sistema de registro (como um banco de dados), assim como exemplificado na Figura 1.

Já Bauer e Bolliet (1968) dizem que *middleware* é o *software* de computador que fornece serviços para aplicações de *software* além daqueles disponíveis pelo sistema operacional.

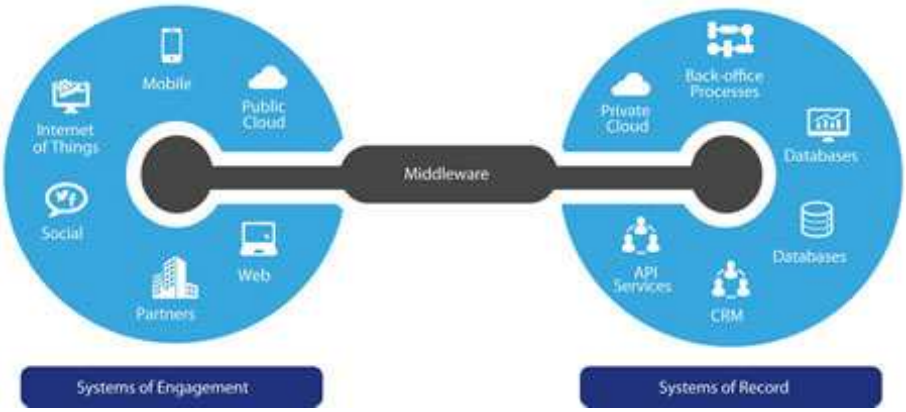


Figura 1 – *Middleware* como intermediário

O que todas essas definições tem em comum é que elas tratam o *middleware* como um mediador entre diferentes *softwares*, gerando uma facilidade na programação final do sistema como um todo.

São amplamente explorados por aplicações distribuídas, sendo a camada que provê a comunicação entre os componentes dos sistemas distribuídos ou até mesmo um serviço adicional (BAUER; BOLLIET, 1968). No contexto deste trabalho, é utilizado na solução do Home Assistant na camada de Home Automation que será explicado na Seção 2.9.1.1

## 2.5 Arquitetura SOA

A arquitetura orientada a serviços (SOA) é uma arquitetura de software que presta serviços entre componentes, por meio de um protocolo de comunicação em uma rede. Seu principal objetivo é trazer a interoperabilidade na aplicação (GOUVEIA; GOUVEIA, 2004), ou seja, não importa quem esteja oferecendo ou consumindo serviço, o hardware ou sistema operacional utilizado, se o protocolo estabelecido for obedecido e se o consumidor e o provedor do serviço estiver na mesma rede, o objetivo será alcançado.

Segundo Erl (2007) e Gouveia (2004) as principais arquiteturas classificadas como SOA são:

SOAP: seu protocolo é baseado no uso de XML para a formação da mensagem;

REST: seu protocolo é baseado nos métodos HTTP (GET, POST, PUT, DELETE);

MOM: sua arquitetura pode ser definida por diferentes protocolos dependendo da sua implementação. O que difere das outras arquiteturas, é que a troca de mensagem é mediada por um *middleware*.

Entre as principais arquiteturas definidas por estes autores, serão utilizadas a REST e a MOM.

### 2.5.1 REST

A arquitetura REST é muito utilizada nos dias atuais por diversas empresas para que aplicações externas possam ter acesso a recuperar (com método GET), atualizar (com método PUT), criar (com método POST) e deletar (com método DELETE) recursos específicos (MASSE, 2012). Essa arquitetura possibilita que outros desenvolvedores criem extensões da aplicação que provê o serviço. Um exemplo é o aplicativo Slack que utilza a API do GitHub possibilitando gerar notificação em um canal do Slack a cada *pull request* submetido em uma *branch* no GitHub, entre outras funcionalidades. No trabalho em questão será utilizado a API da LIFX para recuperar os cenários cadastrados pelo aplicativo deles, e as *playlist* serão recuperadas pela API do Spotify.

### 2.5.2 MOM

A arquitetura *Middleware* Orientado a Mensagem (MOM), é definida por um modelo onde há a troca de unidades autônomas de informação chamadas de "mensagens", onde há uma entidade chamada de *middleware* responsável pelo roteamento e transformação dos dados, ou seja, além de transmitir a mensagem entre o remetente e o destinatário, ele pode transformar a mensagem recebida, caso seja necessário que o destinatário a receba com um conteúdo diferente daquele enviado pelo remetente. Essas mensagens são usualmente enviadas e recebidas de forma assíncrona e não há necessidade que o remetente ou destinatário tenha consciência da existência um do outro (ARTEIRO et al., 2007; KALE, 2015).

## 2.6 *Messager Broker*

Os *message brokers* ou somente *brokers*, são uma das possíveis implementações de *middleware* da arquitetura MOM. Eles além de implementar o roteamento das mensagens implementam uma camada que reduz a complexidade de desenvolvimento de alguma funcionalidade, atendendo a requisitos não funcionais específicos e facilitam a reutilização de funções intermediárias. Por exemplo, um *broker* pode ser usado para gerenciar uma fila de mensagens para vários receptores, fornecendo armazenamento confiável, entrega garantida de mensagens e até mesmo gerenciamento de transações (KALE, 2015). Os *message brokers* podem ser implementados com diversos protocolos diferentes, como o AMQP e o MQTT.

## 2.7 MQTT

O MQTT é um dos possíveis protocolos que se encaixam na arquitetura MOM, projetado para dispositivos restritos e redes de baixa largura de banda, alta latência ou

não confiáveis. Ele é projetado para minimizar a largura de banda da rede sem deixar de atender aos requisitos dos recursos do dispositivo, ao mesmo tempo em que tenta garantir a confiabilidade a um certo grau de garantia de entrega, tornando esse protocolo ideal para tornar recursos no âmbito de IoT acessíveis (ORG, 2014).

Segundo a IBM 2018, a desenvolvedora inicial do protocolo (inicial pois é um software livre, logo qualquer pessoa ou empresa pode contribuir), o MQTT resolve alguns problemas do protocolo HTTP em ambientes de IoT, tais como:

- O HTTP é um protocolo síncrono. O cliente espera que o servidor responda. Os navegadores da web têm esse requisito, mas o custo é a baixa escalabilidade. No mundo da IoT, a comunicação síncrona tem sido um problema devido ao grande número de dispositivos e à rede, muitas vezes não confiável e de alta latência. Um protocolo de mensagem assíncrono é muito mais adequado para aplicativos de IoT. Os sensores podem enviar leituras e permitir que a rede descubra o caminho e a sincronização ideais para entregar aos dispositivos e serviços de destino;
- HTTP é unidirecional. O cliente precisa iniciar a conexão. Em um aplicativo de IoT, os dispositivos e sensores geralmente são clientes, o que significa que eles não podem receber comandos da rede passivamente;
- HTTP é um protocolo de um para um. O cliente faz uma solicitação e o servidor responde. É difícil e caro transmitir uma mensagem a todos os dispositivos na rede, o que é um caso de uso comum em aplicativos de IoT;

Todos esses problemas citados acima, são resolvidos tanto pelo protocolo AMQP quanto pelo MQTT, (como já citado na Subseção 2.6, um dos possíveis protocolos implementados em uma arquitetura MOM). O AMQP é o mais popular no escopo de MOM, porém no ambiente de alto desempenho, a capacidade de computação e a latência da rede geralmente não são uma preocupação. O AMQP é mais utilizados em ambientes corporativos, não sendo adequado para aplicativos de IoT onde há restrição de recursos (IBM, 2018), e para resolver estes problemas, o protocolo MQTT foi inventado.

Tendo em vista que o MQTT é uma boa solução para comunicação entre dispositivos em ambientes de IoT, e que ele tem integração com o Hass.io[6], explicado na Seção 2.9.1, ele pode ser utilizado para fazer a comunicação entre os diversos dispositivos como o sonoff[7] (6.1.2), permitindo controlar motores, e equipamentos de liga e desliga, como lâmpadas.

[6] Transforma um computador tal como o Raspberry Pi em um smart hub equipado com o Home Assistant.

[7] Relé com acesso a WiFi.

## 2.8 IFTTT

O IFTTT fornece uma plataforma de software que conecta aplicativos, dispositivos e serviços de diferentes desenvolvedores para acionar uma ou mais automações envolvendo esses aplicativos, dispositivos e serviços (COMPUTERWORLD, 2018). Ela é baseada na premissa de conectar serviços de IoT fazendo uma integração centralizada, permitindo criar combinações de comandos envolvendo diferentes dispositivos de diferentes fabricantes (COMPUTERWORLD, 2018; BLUELUX, 2018).

Essa plataforma conecta diversos aplicativos e sistemas interessantes para o desenvolvimento desse projeto, como serviços de localização, iluminação e música.

Esse trabalho apresentará alguns casos em que o IFTTT pode ser utilizado em ambientes de automação residencial. Essa tecnologia não será utilizada na solução final do cenário mais completo, mas será mostrado na Seção 4.8.1.2, como ela pode ser integrada ao Home Assistant, fazendo com que ela vire uma ferramenta mais completa, juntamente com o experimento realizado que permite com que a integração entre dispositivo-dispositivo melhore a qualidade de vida de uma pessoa que possui um equipamento de IoT no âmbito residencial, atendendo assim o objetivo da centralização do controle dos dispositivos de IoT em um único aplicativo, definido no presente trabalho.

## 2.9 *Smart Hub*

Um *smart hub* é um dispositivo de hardware que conecta objetos inteligentes em uma rede de IoT e controla as comunicações entre eles (ROUSE, 2018).

Os *smart hub* podem ser proprietários como Alexa da Amazon e a Google Home da Google, ou *smart hub open source*, como Calaos e Home Assistant. Tanto o Calaos quanto o Home Assistant podem ser integrados aos dispositivos Raspberry Pi e o Chip.

### 2.9.1 Home Assistant

O Home Assistant é um *software* escrito em Python que transforma o dispositivo onde ele é instalado em um *smart hub*. Ele pode ser instalado de duas formas diferentes. A primeira forma e a mais usual é utilizar a imagem do Hass.io em uma Raspberry Pi (ou outro dispositivo). Quando ele é instalado com sua própria imagem, há um serviço rodando em *background* que possibilita o acesso de qualquer computador que se encontra na mesma rede, em uma interface web ou em uma interface no dispositivo mobile. A segunda forma de instalação é utilizando o pacote *pip* "homeassistant". Nesta forma é necessário a seguinte linha de comando para levantar o servidor:

```
hass --open-ui
```

A primeira vez em que o servidor é executado, ou que a imagem é inicializada, as dependências do Home Assistant são instaladas em sua versão mais atual. Após a instalação, a interface gráfica fica disponível pelo endereço:

```
IP_DO_RASPBERRY:8123
```

Não há necessidade de configurar um tunelamento de portas.

#### 2.9.1.1 Arquitetura

O Home Assistant permite a centralização da programação de diversos componentes que podem interagir entre si. A arquitetura do *Home Assistant* é representada pela Figura 2.

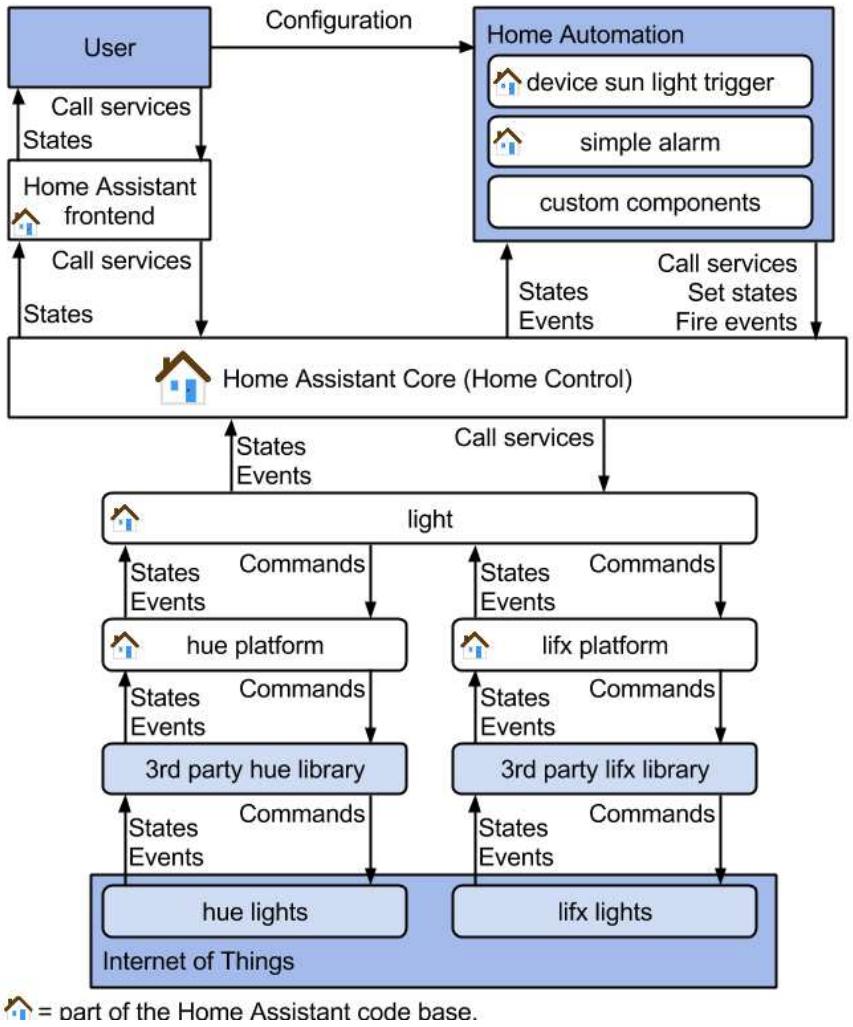


Figura 2 – Arquitetura completa do *Home Assistant*

Os retângulos que possuem o ícone de uma casa, são serviços que se encontram no código fonte do *Home Assistant*. A seguir será detalhado cada elemento representado nessa figura:

**User:** São os usuários do sistema. Eles podem interagir com o *Home Assistant* através de chamadas de serviços utilizando a interface, e podem fazer configuração de componentes na camada de Home Automation;

**Home Assistant Frontend:** É a camada de interface do *Home Assistant*. As chamadas de serviços feitas pelo usuário através do *frontend*, são levadas até a camada de *Home Control*, e também é mostrado para os usuários os estados dos dispositivos de IoT ;

**Home Automation:** É a camada que permite a configuração dos componentes e/ou criação de componentes personalizados. Os elementos de configuração e os componentes personalizados criados neste trabalho, são representados por esta camada. Na Figura 2 são representadas as subcamadas da configuração do gatilho da luz solar

e o alarme que apesar de estarem no código fonte do *Home Assistant*, necessitam de uma configuração dos usuários, e a camada de componentes personalizados, que precisam da implementação do usuário. Este trabalho é implementado em cima de componentes personalizados;

**Home Control:** É a camada core do *Home Assistant*. É responsável por coletar informações e controlar dispositivos, fazendo a centralização de todos os componentes. Nele é implementado a arquitetura MOM 2.6, onde através da camada Home Automation e Home Assistant Frontend, há um disparo de ação via mensagem para o Home Control que coloca essa mensagem em uma fila que será executada a posteriori. Essa mensagem pode ser uma chamada de serviço, um set de um estado ou um disparo de um evento;

**Light:** O componente *light*, na Figura 2, representa todos os componentes que fazem a comunicação entre o *core* e os dispositivos de IoT. Sua implementação é feita através de uma classe (neste caso o nome da classe é "Light") onde são definidas as funções e métodos gerais de comunicação deste tipo de componente. Sua arquitetura é feita seguindo a premissa que o usuário irá configurar o componente através do arquivo "configuration.yaml", podendo assim ter acesso a alteração dos estados, disparar eventos ou chamar um serviço desta camada de componente. Os componentes representados pelas camadas "hue platform"e "lifx platform", são componentes que fazem herança com a classe de comunicação geral, e definem a comunicação específica entre a "3rd party component libary"e o *Home Assistant*. Estas "bibliotecas"normalmente são representadas através de uma API. Neste trabalho serão utilizados os componentes GoogleCast e Spotify que herdam do componente MediaPlayerDevice e o componente LIFX que herda do componente Light e o componente LifxCloudScene que herda do componente Scene. A classe Light, também pode ser chamada de classe Entity, quando representada de forma mais genérica;

**Internet of Things:** É a camada que representa os dispositivos físicos que estão interligados a rede do hass.io. No presente trabalho foram utilizadas as 3 lampadas LIFX, e um aparelho GoogleCast.

#### 2.9.1.2 Definições importantes

**Entity:** Cada dispositivo no Home Assistant é representado como uma entidade. Uma entidade abstrai o trabalho interno do Home Assistant. Um desenvolvedor de um componente integrador, não precisa se preocupar sobre como os serviços ou a máquina de estado funcionam. Em vez disso, ele estende uma classe de Entity e implementa as propriedades e os métodos necessários para o tipo de dispositivo que ele esteja integrando;

**Events:** O core do Home Assistant é impulsionado por eventos. Isso significa que, para reagir a algo que esteja acontecendo em qualquer *entity*, é necessário reagir aos eventos;

**States:** São uma representação atual do objeto *entity*;

**Service:** No fluxo representado pela Figura 2, o usuário faz uma chamada de um serviço pelo o frontend, e o core mostra qual será a camada de *entity* que executará o serviço;

**Commands:** Na figura 2, é uma representação de uma chamada de serviço por uma *entity*, ou seja, o avanço entre as a camadas de serviço de uma *entity* até chegar o comando para o dispostivo físico é através de commands.

#### 2.9.1.3 O arquivo configuration.yaml

Cada componente integrado ao Home Assistant tem uma interface que controla algumas de suas principais funções via arquivo *.yaml*. A princípio não há nenhum componente configurado na interface do Home Assistant (a não ser alguns exemplos em que ele consegue identificar o dispositivo na rede, assim como com o ChromeCast), para isso, é necessário fazer sua configuração pelo arquivo "configuration.yaml". Esse arquivo centraliza toda configuração do *Home Assistant*. Nele também é possível desenvolver diversos componentes, setar valores padrões de interface, localização, segurança, histórico, permitir controle por microfone, criar novos comandos, interfaces, automações e rotinas. Seu formato de arquivo é *.yaml* e são escritos através de dicionários e/ou listas. A seguir serão detalhados os principais elementos que serão utilizados neste trabalho:

**media-player:** É o componente principal de mídia do *Home Assistant*. Ele permite com que os componente que herdam dele tenham o controle do volume, possam ligar, desligar, pausar e dar *play* no dispositivo. O Spotify, e o ChromeCast são componentes de *media-player*. Uma peculiaridade da classe que implementa o componente do Spotify é que ela possui 3 atributos importantes para a solução proposta que podem ser acessados via o estado do componente. São eles: o volume (*float*), o dispositivo (*string*) e o modo aleatório (*boolean*);

**light:** Este componente permite rastrear e controlar várias lâmpadas. A LIFX é utilizada como componente light, sendo necessário passar o IP do servidor e do *broadcast*;

**scene:** Captura estados de entidades e permite que ele seja utilizado a posteriori. Só é possível utilizar uma chamada de um serviço, ou seja, não tem possibilidade de criar uma cena com mais de um tipo de *entity* (*media-player* e *light*, por exemplo). Esse recurso será utilizado configurando a *cloud* da LIFX, ou seja, todos as cenas definidas pelo aplicativo da LIFX, estarão disponíveis para o *Home Assistant*;

**input_number:** Permite setar um valor de entrada inicial, o valor máximo e mínimo e o passo que por sua vez identifica de quanto em quanto a unidade é variada, dando ao usuário a opção de escolher um valor numérico;

**input_select:** Possui um atributo que indica algumas opções, permitindo assim que o usuário a selecione;

**input_boolean:** É uma entrada de dados que permite que usuário escolha entre verdadeiro e falso;

**group:** Grupos permitem que o usuário combine várias entidades em uma. No caso deste projeto, eles são utilizados para unir elementos de *input* na interface para que seus resultados sejam parâmetros de algum *script*;

**sensor:** Os sensores reúnem informações sobre estados e condições. No caso da solução, eles estão sendo utilizados para recuperar os valores inseridos em alguns dos componentes criados de *input*;

**automation:** A automação é dividida em três partes diferentes: um gatilho, uma condição e uma ação. O gatilho descreve eventos que devem acionar a regra de automação, por exemplo, um gatilho pode ser disparado quando um sensor de chuva acusa que irá chover. As condições são testes opcionais que podem limitar a automação apenas para trabalhar em seus casos de uso específicos, por exemplo, se há roupa no varal. A ação será executada quando uma regra for acionada e todas as condições forem atendidas, por exemplo, o painel de proteção a chuva será estendido. Neste caso hipotético, quando o sensor de chuva acenar que irá chover, e se tiver roupa no varal, o painel de proteção contra chuva será estendido. O elemento de automação se assemelha com o funcionamento do IFTTT, e pode ser utilizado em conjunto. No caso deste trabalho, o elemento de automação será utilizado para setar e sincronizar o aleatório para *playlists* no Spotify;

**script:** O componente de script permite que os usuários especifiquem uma sequência de ações a serem executadas pelo Home Assistant quando ativadas. Esse componente será fundamental para a realização deste trabalho, pois com ele será possível criar ações que permitem a seleção de uma *playlist* e para a criação de um componente mais complexo de cenários com múltiplas entidades.

#### 2.9.1.4 Elementos de descentralização de códigos .yaml

Apesar do configuration.yaml ser o centralizador de toda configuração e construção de componentes personalizados do Home Assistant, é possível mudar sua arquitetura utilizando as seguintes opções avançadas:

**!include_dir_list** retornará o conteúdo de um diretório como uma lista, com cada conteúdo de arquivo sendo uma entrada na lista. Não foi utilizado na solução;

**!include_dir_named** retornará o conteúdo de um diretório como um dicionário que mapeia nome do arquivo => conteúdo do arquivo. Foi utilizado na solução para o componente: customize;

**!include_dir_merge_list** retornará o conteúdo de um diretório como uma lista, mesclando todos os arquivos (que devem conter uma lista) em uma lista grande. Foi utilizado na solução para os componentes: sensor, automation;

**!include_dir_merge_named** retornará o conteúdo de um diretório como um dicionário, carregando cada arquivo e mesclando-o em um grande dicionário. Foi utilizado na solução para os componentes: group, script;

**!include** é a declaração que diz ao Home Assistant para inserir o conteúdo de filename.yaml nesse ponto. É utilizado dentro do arquivo "scripts/complete_scenes.yaml"para fazer a reutilização do script de seleção de dispositivo e *playlist* em ambos componentes personalizados criados.

# 3 Metodologia

## 3.1 Tipo de pesquisa

Este trabalho pode ser classificado quanto a natureza sendo uma pesquisa de natureza aplicada, ou quanto ao gênero sendo, uma pesquisa prática, ou seja, é uma pesquisa voltada para intervir na realidade social (PRODANOV; FREITAS, 2013).

> A pesquisa de natureza aplicada objetiva gerar conhecimentos para aplicação prática dirigidos à solução de problemas específicos.
> (PRODANOV; FREITAS, 2013)

## 3.2 Dados obtidos

No trabalho em questão, os dados obtidos são um conjunto de técnicas, ferramentas e sistemas atuantes no mercado que juntos formam uma base sólida que ajuda a construir um novo componente para configuração de cenários em ambientes inteligentes.

Esses dados são apresentados no capítulo de ferramentas **??**, onde é mostrado o detalhamento das escolhas de ferramentas, e o de resultados 4, onde é apresentado o processo de escolha das ferramentas e os conjunto de técnicas e sistemas atuantes no mercado utilizados para a solução final do trabalho.

## 3.3 Forma de obtenção dos dados

A forma de obtenção dos dados é a seleção das amostras, onde é evidenciado os critérios para a sua escolha a qual servirá para a compreensão do objeto de estudo (PRODANOV; FREITAS, 2013).

Para a escolha das ferramentas foram definidos algumas amostras que pudesse resolver o problema e alguns critérios de seleção. A posteriori foi feito uma análise em cima dos critérios adotados onde a escolha de cada ferramenta foi embasada. Toda essa análise é apresentada no capítulo de resultados 4.

## 3.4 Restrições da pesquisa

A pesquisa de ferramentas (como Raspberry Pi, Sonoff e etc.), sistemas de *media player* e iluminação (como Spotify, Lifx e etc.) e técnicas será limitada a aquelas integradas

com o sistema escolhido *open source* de automação residencial, pois o foco desse trabalho não é criar novas ferramentas, sistemas e/ou técnicas, e sim utilizar as que já estão no mercado para a construção de um componente de cenários mais robusto.

# 4 Resultados

Esse capítulo se trata dos resultados da pesquisa.

## 4.1 Escolha do software de automação residencial

Com o fim de escolher um software de que possa gerenciar os dispositivos de automação residencial *open source*, foram analisados alguns encontrados em pesquisas realizadas. Após um filtro inicial foram separados os softwares: Uiot[1], Calaos[2], Domoticz[3] e o Home Assistant[4], nos quais avaliou-se os seguintes aspectos:

**Possui fórum de dúvidas:** atributo binário (tem ou não);

**Língua principal do fórum:** preferência por português, depois inglês;

**Licença:** preferência por licença permissiva;

**Apoio da comunidade:** nota 1-5 sendo 5 a mais alta;

**Nível de documentação:** nota 1-5 sendo 5 a mais alta;

**Site web:** atributo binário (tem ou não);

**Mobile:** atributo binário (tem ou não);

**Interface traduzida para português:** atributo binário (tem ou não);

**Interface traduzida para inglês** atributo binário (tem ou não);

**Integração com Lifx:** atributo binário (tem ou não);

**Integração com Google Cast:** atributo binário (tem ou não);

**Integração com Sonoff:** atributo binário (tem ou não);

**Integração com Spotify:** atributo binário (tem ou não);

**Facilidade para evolução de cenários:** nota 1-5 sendo 5 a mais alta.

[1] Uiot - consulte: <https://uiot.org/>
[2] Calaos - consulte <https://calaos.fr/en/>
[3] Domoticz - consulte: <http://www.domoticz.com/>
[4] Home Assistant - consulte: <https://www.home-assistant.io/>

Os critérios binários, foram analisados de acordo a existência ou não do critério. Os critérios de nota, foram atribuídos de forma pessoal, onde foi feito uma análise do critério, sendo analisando o código fonte ou a documentação referente ao critério. Os de preferência foram feitos de forma pessoal. Sendo assim, o Quadro 1 representa a análise feita para a escolha do software:

Quadro 1 – Análise sobre as ferramentas

| **Ferramenta** | Home Assistant | Domoticz | Calaos | UIOT |
|---|---|---|---|---|
| **Fórum de Dúvidas** | Sim | Sim | Sim | Não |
| **Língua Principal do Fórum** | Inglês | Inglês | Francês | - |
| **Licença** | MIT | GPLv3 | GPL | GPL |
| **Apoio da Comunidade** | 5 | 4 | 3 | 1 |
| **Nível de Documentação** | 5 | 5 | 3 | 1 |
| **Site Web** | Sim | Sim | Sim | Sim |
| **Mobile** | Sim | Sim | Sim | Não |
| **Interface traduzida para português** | Sim | Sim | Não | Sim |
| **Interface traduzida para o inglês** | Sim | Sim | Sim | Não |
| **Integração com o Lifx** | Sim | Sim | Não | Não |
| **Integração com o Google Cast** | Sim | Não | Não | Não |
| **Integração com o Sonoff** | Sim | Sim | Não | Não |
| **Integração com o Spotify** | Sim | Sim | Não | Não |
| **Facilidade para evolução de cenários** | 4 | 2 | 1 | 1 |

Tendo em vista o quadro apresentado, o software que melhor se encaixou nos critério adotados foi o Home Assistant.

## 4.2 Escolha da lâmpada inteligente

Para escolher a lâmpada inteligente, foram analisados as duas mais conhecidas do mercado, a Hue[5] e a Lifx[6]. Ambas aceitam o protocolo IFTTT, possuem uma API que disponibiliza a interação de cenários cadastrados na nuvem e ambas já possuem integração com o Home Assistant, logo a escolha pela Lifx, foi definida por já ter um componente no Home Assistant que faz a tradução de cenários da nuvem da Lifx. Mesmo que esse componente tenha que ser adaptado ao contexto desse trabalho, já é uma vantagem em cima da Hue.

## 4.3 Escolha do software para a seleção de *playlist*

A escolha do software para a seleção de *playlist* foi feita levando em consideração a análise feita por Iqbal (2019), que diz que a maior plataforma de *stream* de música em números de inscrito é o Spotify, além de que há uma familiaridade no uso da ferramenta por parte do desenvolvedor do projeto e suas possíveis integrações via API.

[5] Hue - consulte: <https://www2.meethue.com>
[6] Lifx - consulte: <https://www.lifx.com/>

## 4.4 Solução

Esta Seção apresentará a evolução da solução, disponibilizada no repositório <https://github.com/arturbersan/configuration_ha.>

### 4.4.1 Definição de uma nova arquitetura

Como explicado na Seção 2.9.1.3, o arquivo "configuration.yaml"centraliza toda a criação de novos componentes na camada de Home Automation. No projeto em questão foi percebido a necessidade de separar o código em novas camadas além do "configuration.yaml", para uma melhor organização.

Essa necessidade foi percebida por conta de dois fatores. O primeiro fator foi a necessidade de um reaproveitamento de código, onde a arquitetura padrão do arquivo "configuration.yaml"não permite esse tipo de ação. Um segundo motivo, foi a necessidade de separar em várias pastas toda configuração e código. O Home Assistant por padrão permite que cada componente que esteja sendo utilizado no arquivo "configuration.yaml", seja separado em um arquivo .yaml, mas quando há um trecho de código maior de um mesmo componente, esta solução começa a não atender muito bem, pois começa a ter poluição do código.

Utilizando as técnicas apresentadas na Seção 2.9.1.4, foram definidos a seguintes camadas:

**groups:** onde é definido todos os componentes tipo group;

**automations:** onde é definido todos os componentes tipo automation;

**scripts:** onde é definido todos os componentes tipo script;

**sensors:** onde é definido todos os componentes tipo sensor;

**customizes:** onde é definido todos os componentes tipo customize;

**data_template_script:** onde é definido todos os corpos de scripts. Por esta pasta ter sido criada, é possível reaproveitar o código de um script em mais de um componente personalizado.

### 4.4.2 Correlação entre a arquitetura definida e a arquitetura do Home Assistant

A figura a seguir é uma adaptação da imagem 2, no contexto do projeto, juntamente com as interações feitas com a nova arquitetura na camada de Home Automation:

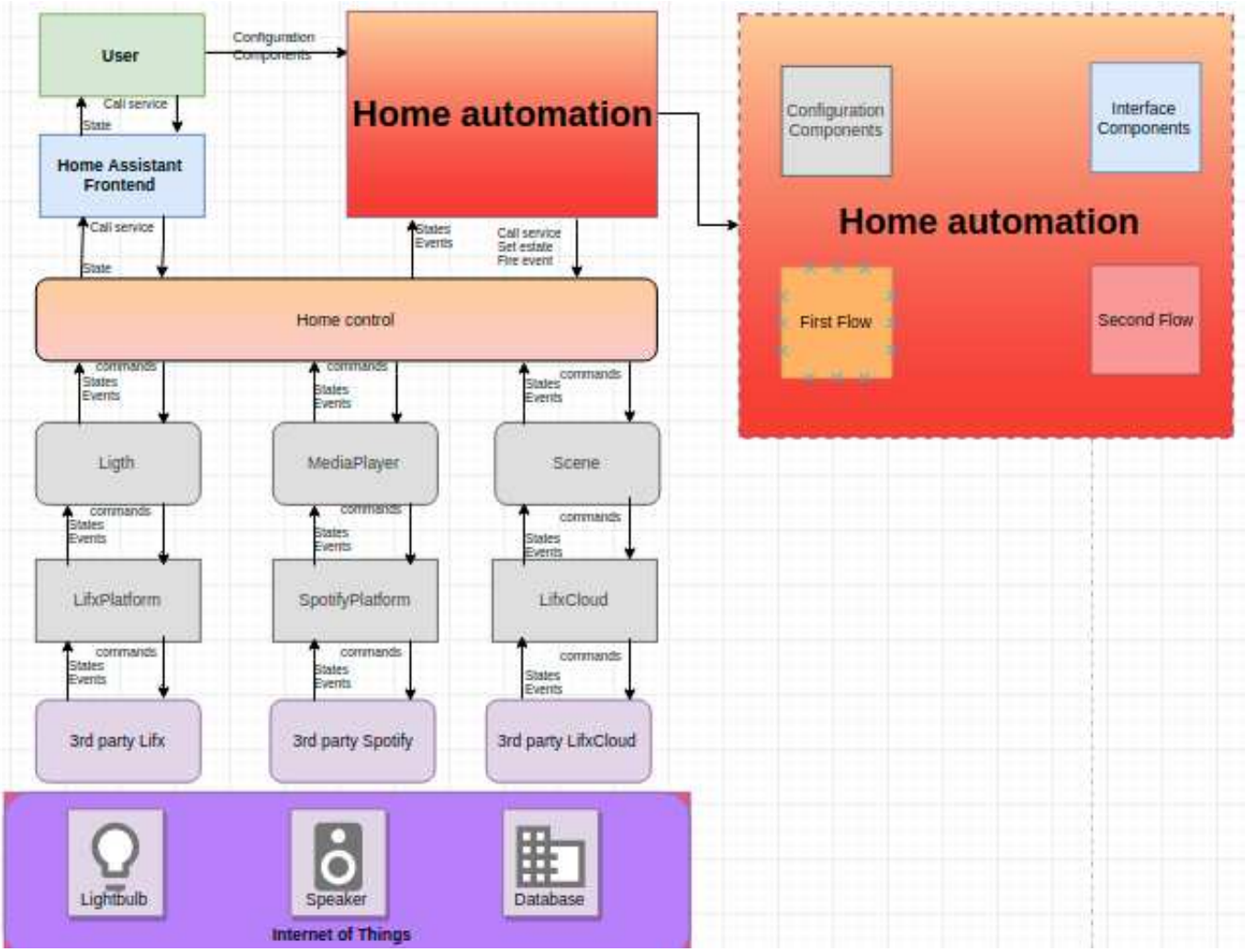


Figura 3 – Diagrama geral da solução

Nesta imagem há algumas diferenças quando comparada com a imagem 2. A primeira é que o retângulo representado como "Light"na imagem 2, é uma representação geral de uma Entity, já nos retângulos correlacionados desta imagem possui todas representações de Entities utilizadas na solução(Light - LifxPlatform, MediaPlayer - SpotifyPlatform, Scene - LifxCloud). Uma segunda diferença entre as imagens é que na 3 há uma representação geral de 4 fluxos que estão dentro da camada Home Automation, nesta solução.

A explicação detalhada de cada fluxo serão apresentas nas sub seções seguintes.

#### 4.4.2.1 Configuration Components

Nesta seção será mostrado as configurações feitas para os componentes selecionados da camada de Entity utilizados na solução. O único dispositivos de IoT que não foi necessário fazer alguma configuração no arquivo "configuration.yaml"do Home Assistant foi o ChromeCast, pois esta comunicação é estabelecida de forma automática. Para que

isso fosse possível, foi necessário que a Raspberry e o ChromeCast estivessem na mesma rede.

Para configurar o ChromeCast a rede, foi necessário seguir os seguintes passos:

Encaixa-lo na entrada HDMI de uma televisão;

Baixar o aplicativo do ChromeCast;

Fazer a configuração de rede pelo aplicativo baixado.

Sendo assim, logo após a primeira inicialização do Hassio, podemos acessar o dispositivo do ChromeCast através da interface do Home Assistant. As imagens seguintes mostram a disposição deste componente na interface do Home Assistant:

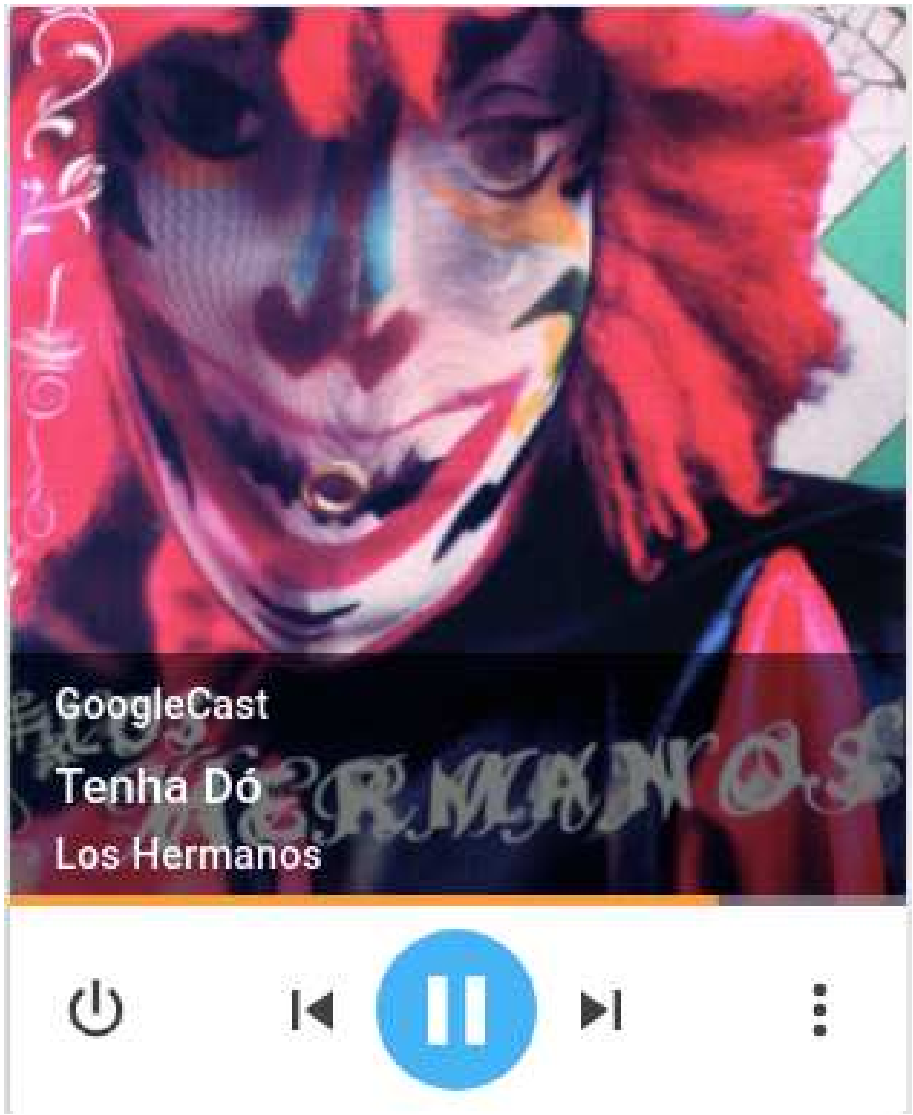


Figura 4 – Chrome Cast

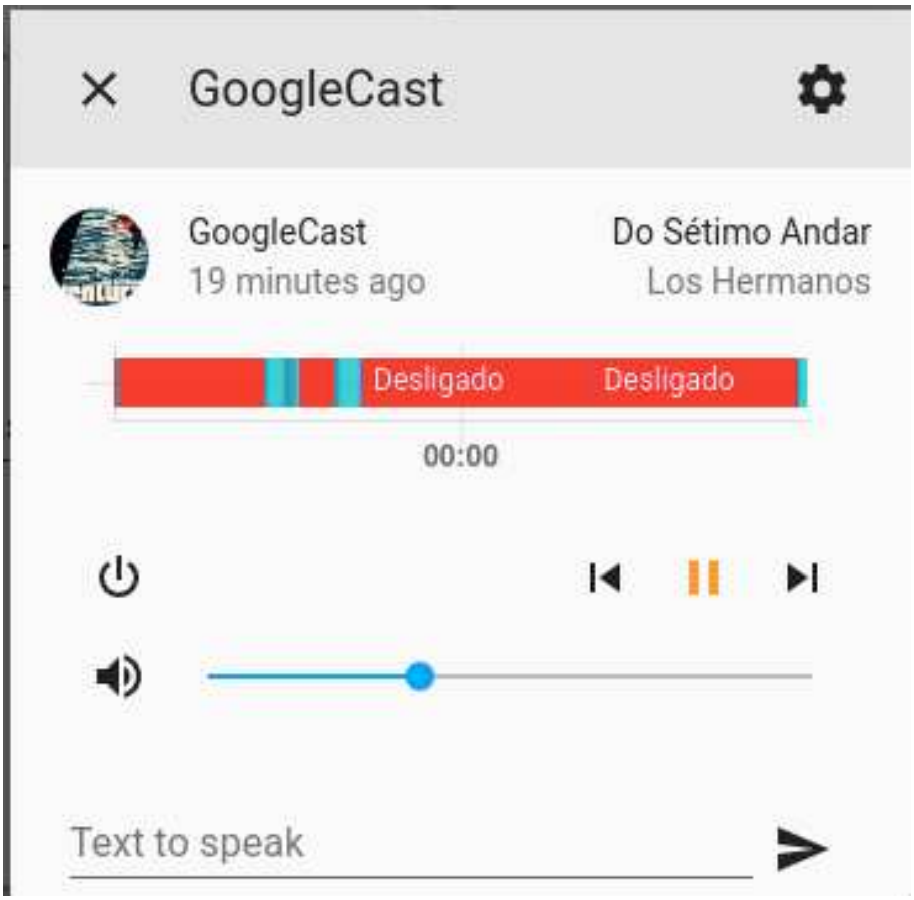


Figura 5 – Chrome Cast Modal

A representação do fluxo Configuration Components apresentado na Figura 3, pode ser apliado pela imagem a seguir:

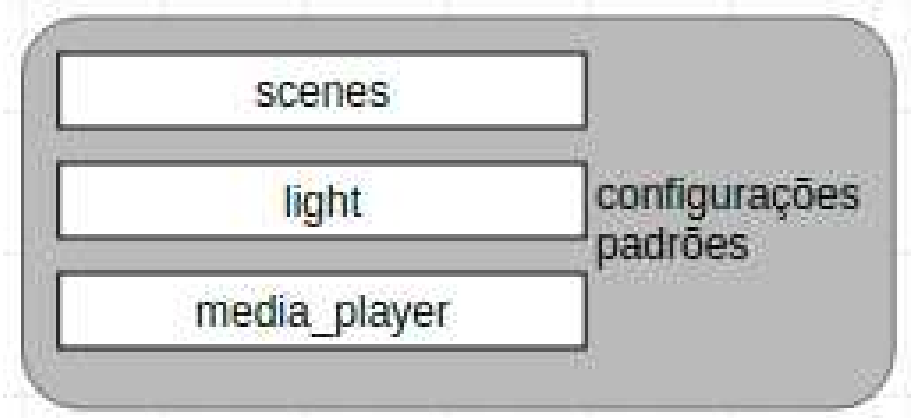


Figura 6 – Configuration Components

Toda camada de Configuration Components é apresentada dentro do Home Assistant foi feita pelo arquivo "configuration.yaml". Para cada retângulo com fundo branco representado nesta imagem, foi necessário a configuração da API externa referente a Entity em questão, juntamente com a configuração dentro do Home Assistant. Sendo assim,

ao realizar cada uma destas configurações, foi possível acessar os componentes Entity apresentado na Figura 3, pela interface do Home Assistant e dentro de componentes personalizados.

#### 4.4.2.1.1 Configuração do componente LIFX

Para configurar o componente LIFX, primeiro foi necessário conectar a lampada ao roteador que provê Internet via WiFi para a Raspberry. Para isso foi necessário seguir os seguintes passos:

Ligar a lampada em uma fonte. No caso desta solução foram feitos alguns testes com uma luminária simples e depois as lampadas foram colocadas em diferentes *plafons*[7];

Baixar o aplicativo do LIFX na *play store* (aplicativo que permite usuários baixem aplicativos para o sistema IOS);

Conectar o *smartphone* a mesma rede do roteador que provê Internet via WiFi para a Raspberry;

Conectar a lampada a mesma rede do roteador nas opções de configuração do *smartphone*.

Este processo foi feito para 3 lampadas da marca LIFX. Os nomes das lampadas foram definidas como: "Cama", "Guarda roupa", "Mesa de trabalho".

O próximo passo foi colocar as seguintes linhas de configuração no arquivo "configuration.yaml"para que o Home Assistant consiga interagir com a LIFX:

```
light:
  - platform: lifx
    server: 192.168.0.17
    broadcast: 192.168.0.255
```

Após essas configurações foi possível acessar as lampadas da LIFX através da interface do Home Assistant, como demostra a figura a seguir:

[7] Luminária de formatos e materiais diversos que é colocada junto ao teto.


Figura 7 – Iluminação

#### 4.4.2.1.2 Cenários de iluminação

O componente Scene (Cenário), no Home Assistant normalmente é definido como um componente personalizado dentro da camada de Home Automation, onde o estado de um objeto Entity (O termo objeto, é referente a orientação objetos, ou seja, neste caso é a instancia de uma classe Entity) é capturado para um uso posterior. Neste caso padrão, cada definição de cenário tem que ser programado através do arquivo "configuration.yaml".

O conceito de cenários em dispositivos de lampadas também é utilizado pelo aplicativo da LIFX, sendo que sua programação é muito mais intuitiva do que quando levamos em comparação a programação pelo hassio. Isso acontece pois no app da LIFX, há uma seleção de cor e brilho em tempo real e o usuário salva os estados do conjunto de lampadas utilizando uma interface onde é possível transitar facilmente entre uma cor e outra e seu brilho. Na definição de cenários feitas através do hassio é necessário entender a sua estrutura e fazer sua programação utilizando os elementos de explicados na Seção **??**, além das cores serem definidas em RGB e o brilho ser definido por um numero inteiro e, para conferir o resultado de um cenário é necessário fazer a compilação do código.

Ao configurar as 3 lampadas LIFX foi possível criar diversos cenários pelo próprio aplicativo, tais como:

**Escolher roupa:** Guarda roupa e Mesa de trabalho ligadas com cores claras com um brilho alto;

**Festa:** Cama, Guarda roupa e Mesa de trabalho ligadas com cores escuras com brilho alto;

**Filme:** Mesa de trabalho ligada com um amarelo claro e com o brilho bem baixo;

**Relaxante:** Mesa de trabalho, Cama e Guarda roupa ligadas com cores claras e brilho médio;

**Romantico:** Mesa de trabalho, Cama e Guarda roupa ligadas com cores escuras e brilho baixo;

**Trabalhar de dia:** Mesa de trabalho ligada com uma cor clara e brilho alto.

Ao criar esses cenários no aplicativo da LIFX, eles foram armazenados na LIFX Cloud, possibilitando recuperá-los e utilizá-los pela interface do Home Assistant. Para isso foi necessário seguir os seguintes passos:

Logar na conta em que eles foram configurados;

Gerar um novo *token* de acesso a conta;

Adicionar o *token* no arquivo "configuration.yaml".

Segue abaixo a configuração de cenários no arquvio "configuration.yaml":

```
scene:
  - platform: lifx_cloud
    token: YOUR_LIFX_TOKEN
```

Desta forma, é possível utilizar os mesmos cenários definidos pelo aplicativo da LIFX pela interface do Home Assistant, como é demonstrado na Figura abaixo:

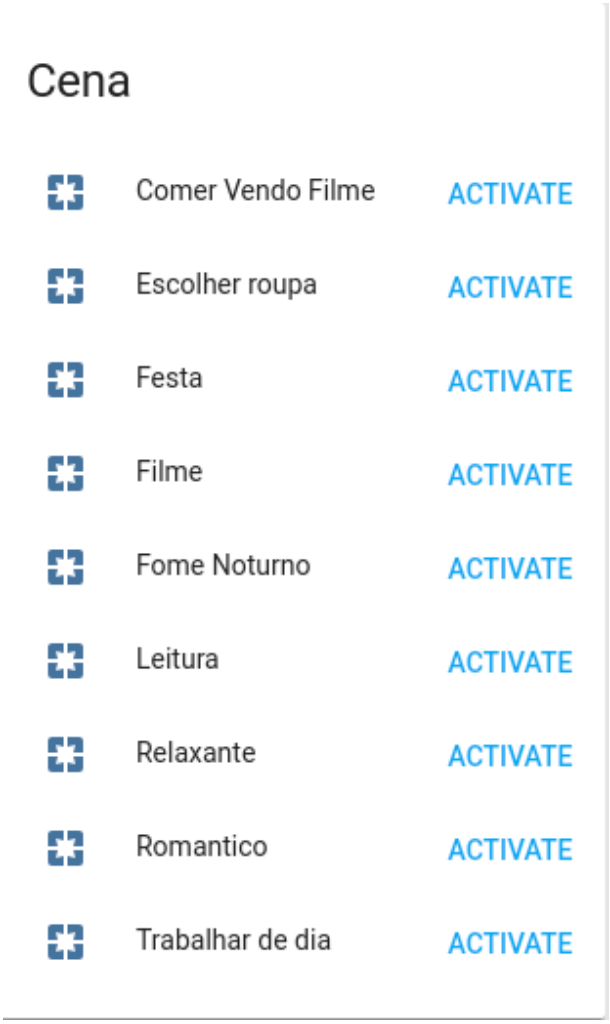


Figura 8 – Cenários LIFX

#### 4.4.2.1.3 Spotify

Para utilizar o Spotify juntamente com o Home Assistant foi preciso seguir os seguintes passos:

Criar uma assinatura paga;

Logar como desenvolvedor do Spotify;

Criar uma nova aplicação Spotify. Essa fase permite que as funcionalidades da API do Spotify sejam disponíveis pela para o conta que foi logada;

Adicionar os *tokens* gerados no arquivo "configuration.yaml".

Segue abaixo a configuração do Spotify:

```
media_player:
  - platform: spotify
    client_id: YOUR_CLIENT_ID
    client_secret: YOUR_CLIENT_SECRET
```

Após essas configurações foi possível acessar o Spotify pela interface do Home Assistant, como mostra a figura a seguir:

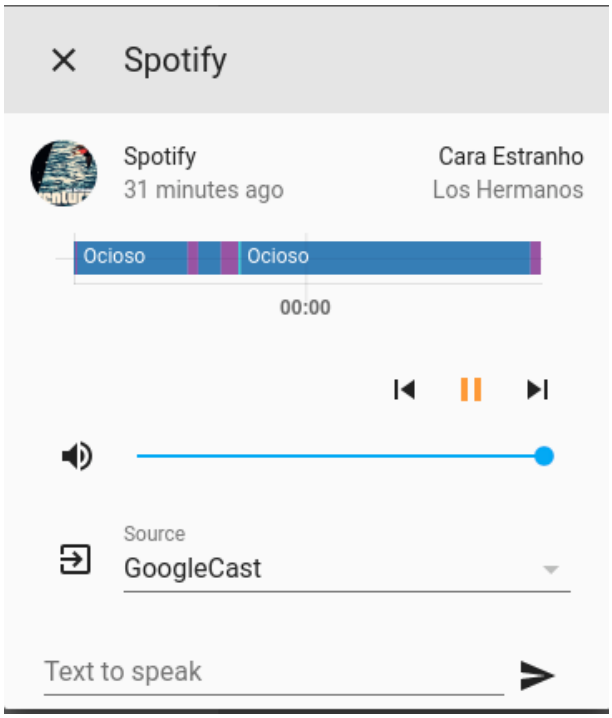


Figura 9 – Spotify Modal

#### 4.4.2.2 Interface Components

Nesta Seção serão apresentados todos os componentes de interface criados e sua integração geral com a arquitetura definida na Figura 3.

Primeiro ponto importante, é que todo componente definido nessa Seção, tem sua definição feita na camada de Home Automation e dentro do código fonte do Home Assistant ele é transformado em um componente de Entity, logo é possível acessar seus *states*. A segunda característica importante para a solução, é que esses componentes possui uma representação na interface do Home Assistant, ou seja, tem uma representação dentro da camada Home Assistant Frontend.

A Figura a seguir é a representação geral destes componentes:

##### 4.4.2.2.1 Os *inputs*

Foram feitos 5 componentes de *input*, são eles:

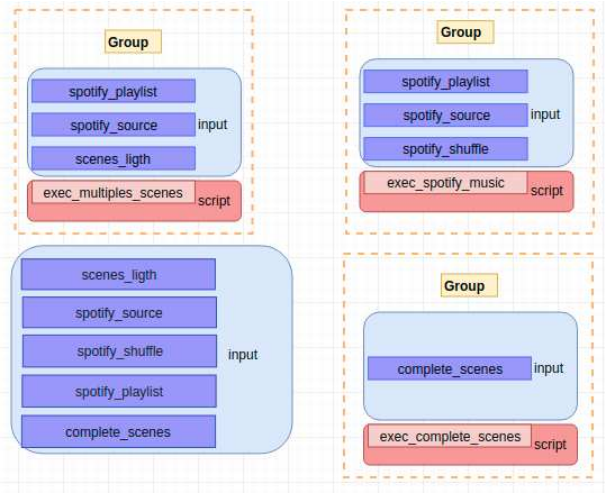


Figura 10 – Interface Components

**O spotify_playlist:** um input_select que representa uma lista de *playlists* do Spotify;

**O scene_light:** um input_select que representa uma cenários de iluminação;

**O spotify_shuffle:** um input_boolean que representa se uma *playlist* é aleatória ou não;

**O spotify_source:** um input_select que representa uma lista de dispositivos que podem executar o Spotify;

**O complete_scenes:** um input_select que representa uma lista de cenários que controla diversos dispositivos.

Esses componentes são utilizados como parâmetro para a execução de *scripts*, e por isso são agrupados em conjunto, fazendo parte do Second Flow definido na Figura 3. Outro uso desses componentes é em conjunto com os *automations*, que por sua vez fazem parte do First Flow.

Após a criação de um componente *input*, sua instancia é criada, permitindo assim, que a máquina de estados e os serviços definidos por sua classe Entity possa ser acessado por outros componentes.

#### 4.4.2.2.2 Os *groups*

Foram feitos 3 componentes de *group*, são eles:

**O spotify_playlist:** agrupa os componentes de *input* spotify_playlist, spotify_shuffle e spotify_source juntamente com a execução do *script* "spotify_playlist.yaml";

**O multiples_scenes:** agrupa os componentes de *input* spotify_playlist, scene_light e spotify_source juntamente com a execução do *script* "multiples_scenes.yaml";

**O complete_scenes:** agrupa o componente de *input* complete_scenes juntamente com a execução do *script* "complete_scenes.yaml".

Após a criação destes componentes foram gerados os seguintes elementos no Home Assistant Frontend:

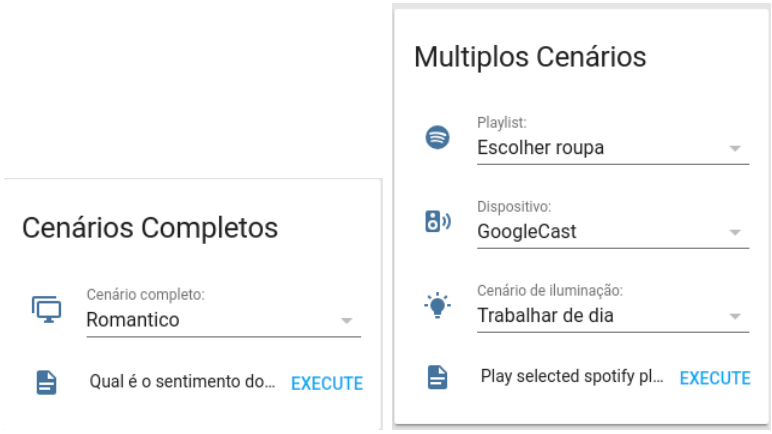


Figura 11 – Cenários Completos / Múltiplos

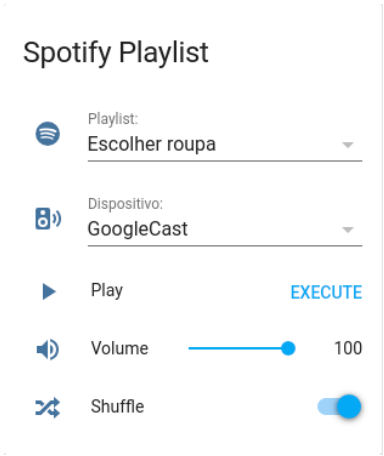


Figura 12 – Spotify Playlist

#### 4.4.2.3 First Flow

O First Flow é a representação completa do funcionamento dos componentes de *automations* definidos na solução. Primeiro ponto importante a ser levado em consideração é que esses componentes, não fazem parte da solução de cenários completos, mas são muito importantes para a validação da proposta de valor deste projeto, pois eles são fundamentais para o controle geral de algumas funcionalidades dos dispositivos de IoT,

ou seja, eles são necessários para fazer com que o aplicativo do Home Assistant possa ter o controle mais próximo das funcionalidades que cada aplicativo possui individualmente, diminuindo assim o uso de diversos aplicativos para o controle de uma casa inteligente. As funcionalidades entregues por esses conjuntos de componentes são:

**O controle do volume do Spotify:** essa funcionalidade não funciona juntamente com o ChromeCast, pois o ele não permite a alteração do volume do dispositivo através do Spotify. Esse problema também acontece com o aplicativo do Spotify, sendo que o podemos resolve-lo através do Home Assistant, utilizando o dispositivo individual do ChromeCast representado pela Figura 5. Apesar disso, essa funcionalidade permite o controle do volume em outros dispositivos como um computador pessoal, uma caixa de som ou um celular;

**O controle dos dispositivos que tocam o Spotify:** Essa funcionalidade permite com que o usuário selecione qualquer dispositivo que esteja conectado na mesma rede do Raspberry;

**O controle do *shuffle* no Spotify:** O *shuffle* (ou aleatório no português), permite com que usuário do Home Assistant selecione a mídia que está em execução (pode ser uma *playlist*, um álbum, um *podcast* e etc.), seja tocada no modo aleatório.

Sua representação arquitetural pode ser representada pela seguinte Figura:

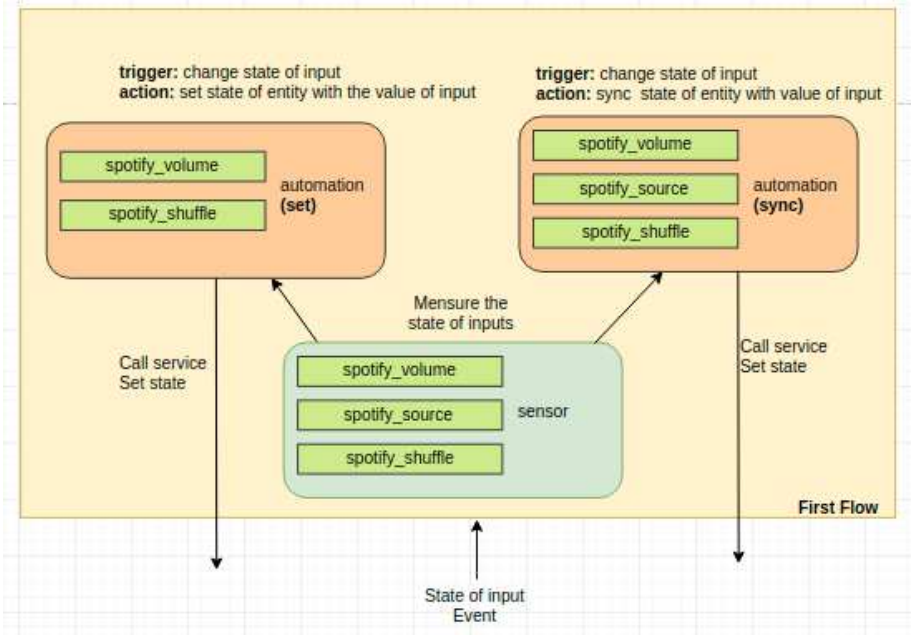


Figura 13 – First Flow

Para a compreensão deste fluxo é necessário entender que a Figura 13 está dentro de uma representação maior apresentada pela Figura 3, onde a seta que entra na caixa

representada pelo Home Automation é a mesma representação da seta que entra na caixa do First Flow, e a seta que sai da caixa do Home Automation é a mesma seta que sai da caixa First Flow.

Como é possível perceber, o First Flow possui 3 etapas representadas pelas setas presentes na Figura 13. São elas: o disparo de um *event*, a medição de um *state*, e o *automation*.

#### 4.4.2.3.1 O disparo de um *event*

Essa etapa é o ponto de partida para o First Flow iniciar, onde o usuário do Home Assistant através do Home Assistant Frontend muda o *state* de um componente *input*, disparando assim um *event* que vai servir como entrada da próxima etapa do fluxo.

Um exemplo de um disparo de um *event* é quando o usuário clica no botão "*Shuffle*"apresentado na figura 12.

Uma segunda forma de disparar um *event*, é quando ocorre uma mudança de *state* de algum atributo mapeado pelo Home Assistant, assim como a mudança do dispositivo que toca o Spotify que acontece sem a interferência de algum usuário através do Home Assistant.

#### 4.4.2.3.2 A medição de um *state*

Como foi explicado na seção anterior, o First Flow, pode ser inicializado por dois disparos destintos de um *event*. O segundo disparo explicado é o ponto importante para essa etapa, pois quando há uma mudança de *state* do objeto Entity do Spotify por uma ação não mapeada no Home Assistant, há uma necessidade de medir o valor deste *state* para que o componente de *input* que representa este *state*, possa ser atualizado assim então, o usuário do Home Assistant consegue ter um *feedback* do que está acontecendo no dispositivo do Spotify.

A etapa de medição de um *state* é feita pelo componente do tipo sensor. Foi necessário criar 3 componentes do tipo sensor, são eles:

**Spotify Volume:** mede o valor atual do *state* de volume do objeto Entity do Spotify.

**Spotify Shuffle:** mede o valor atual do *state* de *shuffle* do objeto Entity do Spotify.

**Spotify Source:** mede o valor atual do *state* do dispositivo que está tocando a mídia do Spotify.

#### 4.4.2.3.3 Os *automations*

Os *automations* definidos na solução foram separados em dois tipos, os *automations* de "*set*", e os de "*sync*". Os *automations* de "*set*"podem ser definidos da seguinte forma:

**Requisito a ser atendido:** Esse tipo de *automation* tem o papel de setar o valor do *state* do dispositivo Spotify;

**Os *triggers*:** Os *triggers* desse tipos de dispositivos é mapeado quando há uma mudança do valor do *state* de um componente do tipo *input*. Por exemplo, quando o usuário clica no botão "*Shuffle*"apresentado na figura 12, ou seja, é mapeado o disparo de um *event* do tipo *input*;

**As *actions*:** As ações deste componente é setar o valor mapeado pelos *triggers* com o valor atual do *state* do componente do tipo *input* que teve seu valor alterado;

**Necessidade de *delay*:** Não há necessidade de *delay*;

**Tempo de resposta:** O tempo médio de resposta desse tipo de *automations* é entorno de 6 segundos, considerando desde o momento em que o usuário faz a ação através da interface do Home Automation até o momento que a ação é executada afetando assim o dispositivo em questão.

Os *automations* de "*sync*"podem ser definidos da seguinte forma:

**Requisito a ser atendido:** Esse tipo de *automation* tem o papel de sincronizar o valor do *state* do dispositivo Spotify, com o valor do input que representa o seguinte *state*. Podemos entender como o caminho inverso das *automations* do tipo "*set*", pois enquanto os *automations* do tipo "*set*"altera o valor do *state* do objeto de Entity do Spotify, os do tipo "*sync*"muda o valor de algum *state* do tipo *input* quando há alguma alteração do valor do *state* do objeto de Entity do Spotify;

**Os *triggers*:** Os *triggers* desse tipos de dispositivos são mapeados quando há uma mudança do valor do *state* do *source*, ou do volume ou do *shuffle* do Spotify, sem a interferência do uso do Home Assistant. Por exemplo, através do aplicativo do Spotify o usuário muda o dispositivo do Spotify para o ChromeCast. Para conseguir mapear o valor do *state* do objeto de Entity do Spotify, é necessário o uso dos sensores criados;

**As *actions*:** As ações deste componente é setar o valor dos *state* dos componentes do tipo *input* mapeado pelos *triggers* com o valor atual do *state* do objeto de Entity do Spotify;

**Necessidade de *delay*:** Esse componente há uma necessidade de um *delay*, pois há uma possibilidade da informação mudar, e com o *delay*, há uma garantia maior para que a *automation* funcione de acordo com o esperado. *triggers* com o valor atual do *state* do objeto de Entity do Spotify;

**Tempo de resposta:** O tempo de resposta depende de alguns fatores. O primeiro fator é que ele necessita do uso de sensores, e o Home Automation, faz a atualização do sensor no intervalo de 30 segundos, sendo que esse delta do tempo afeta o tempo de resposta desse *automation*. O segundo fator é *delay* de 5 segundos que é necessário para o bom funcionamento, sendo que esse fator é fixo. O último fator é o tempo de resposta do próprio componente que é entorno de 6 segundos. Desta forma, o tempo de resposta deste componente gira em torno de 11 segundos e 41 segundos.

### 4.4.3 Second Flow

O Second Flow é a representação da execução dos *scripts* e pode ser definida pela seguinte Figura:

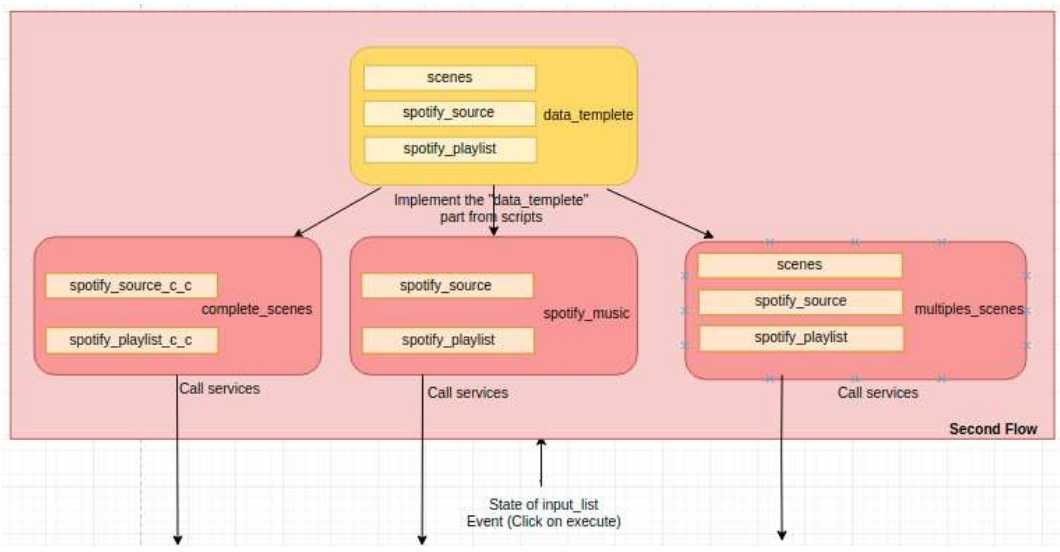


Figura 14 – Second Flow

Assim como no First Flow, a representação do Second Flow está dentro de uma representação maior da Figura 3.

Primeiramente vale relembrar que o Home Assistant possui uma implementação de cenários, mas ela é limitada a configuração de somente um tipo de serviço. A solução deste trabalho visa a configuração de cenários que une a utilização de múltiplos serviços. Será apresentado como foi feita a união do serviço de cenários de iluminação do próprio Home Assistant juntamente com a seleção de uma *playlist* específica do Spotify. A posteriori, será mostrado como um novo serviço pode ser adaptado a solução feita.

A interface do Home Assistant Frontend funciona com o *layout* de *cards* que junta funcionalidades em comum que são agrupadas pelos componentes de *groups* descritos na Seção 4.4.2.2.2. Os elementos de *input* agrupado em cada *group* é a representação do *state* que da o inicio do Second Flow. O *event* definido nesse fluxo é a execução de uma rotina que pega os *states* dos componentes *input* como parâmetro para representar a funcionalidade principal do *script*. O corpo do *script* é separado em outra camada (data_template), para que este trecho de código possa ser utilizada em diferentes *scripts*. Por fim, o *call service* é o resultado final da execução do *script*.

## 4.5 Implementação do componente Spotify Playlist

O *card* Spotify Playlist representa a Figura 12 e possibilita a execução das seguintes funcionalidades:

**Seleção de playlist:** Funcionalidade que faz parte da execução do *script* "spotify_music"definido na Figura 14;

**Seleção de dispositivos:** Funcionalidade que faz parte da execução do *script* "spotify_music"definido na Figura 14;

**Seleção de volume:** Funcionalidade que faz parte da execução do First Flow na Figura 13;

**Seleção de *playlists* aleátorias ou não:** Funcionalidade que faz parte da execução do First Flow na Figura 13.

Para criar o *card* Spotify Playlist e deixá-lo funcional, foi necessário criar os componentes "input_select", "input_boolean"e o "input_number", além de criar um novo grupo, algumas automações, um *script*, alguns sensores e novas customizações.

O *group* utilizado para criar um novo *card* une os *inputs* de seleção de *playlist* e dispositivos executando duas *calls service* com os parâmetros selecionados fazendo parte do Second Flow. O *input* numérico que representa o volume e o *input* booleano que indica se é uma *playlist* aleatória ou não fazem parte do First Flow.

### 4.5.1 Seleção de Playlist

A seleção de *playlist* funciona da seguinte maneira:

**O input:** O *input* de seleção de *playlist* é um componente que permite a seleção dos seus nomes;

**O ícone:** Possui o ícone do Spotify;

**O data_template:** O data_template desta funcionalidade recebe o *input* como parâmetro e retorna o endereço da execução da *playlist* no Spotify, selecionando a Entity media_player.spotify;

**O script:** O *script* faz a *call service* da execução da mídia selecionando a Entity media_player.play_media passando a referência da *playlist* no Spotify.

### 4.5.2 Seleção de Dispositivos

A seleção do dispositivo funciona da seguinte maneira:

**O input:** O *input* de seleção do dispositivo permite a seleção do dispositivo que irá tocar a mídia

**O ícone:** Possui o ícone de WiFi;

**O data_template:** O data_template desta funcionalidade recebe o *input* como parâmetro e retorna o endereço da execução do dispositivo no Spotify, selecionando a Entity media_player.spotify;

**O script:** O *script* faz a *call service* da execução da mídia selecionando a Entity media_player.select_source passando o nome do dispositivo no Spotify.

### 4.5.3 Seleção de Volume

A seleção do volume funciona da seguinte maneira:

**O input:** O *input* numérico de volume que varia de 0 a 100 em um passo de 1 em 1;

**O ícone:** Possui o ícone de volume;

**As automations** Suas *automations* atualiza o volume do Spotify quando o *input* é alterado, ou atualiza o *input* quando o volume do do Spotify muda sem a interferência do Home Assistant;

### 4.5.4 Seleção de Aleatoriedade

A seleção de aleatoriedade funciona da seguinte maneira:

**O input:** O *input* é booleano;

**O ícone:** Possui o ícone de variância de aleatoriedade;

**As automations** Suas *automations* atualiza o modo aleatório do Spotify quando o *input* é alterado, ou atualiza o *input* quando o modo aleatório do do Spotify muda sem a interferência do Home Assistant;

## 4.6 Implementação do componente Múltiplos Cenários

O *card* Múltiplos Cenários representa a imagem da direita da Figura 11 e possibilita a execução das seguintes funcionalidades:

**Seleção de playlist:** Funcionalidade que faz parte da execução do *script* "multiple_scenes"definido na Figura 14;

**Seleção de dispositivos:** Funcionalidade que faz parte da execução do *script* "multiple_scenes"definido na Figura 14;

**Seleção de cenários de iluminação:** Funcionalidade que faz parte da execução do *script* "multiple_scenes"definido na Figura 14;

Para criar o *card* Múltiplos Cenários e deixá-lo funcional, foi necessário criar o elemento "scenes_light"dentro do componente "input_select"e reaproveitar os de "input_select"utilizados no *card* Spotify Playlist, além de criar um novo grupo e uma nova customização.

O grupo criado une os mesmos *inputs* de seleção do *card* Spotify Playlist e acrescenta o *input* de seleção de cenário de iluminação, juntamente com o *script* "multiple_scenes"executando as mesmas *calls service* do Spotify Playlist além da *call service* de seleção de cenário de iluminação fazendo parte do Second Flow.

O maior detalhe deste componente é que a seleção de *playlist* e do dispositivo são chamadas de serviço, ou seja, não são uma representação de cenários, diferentemente da seleção de cenários de iluminação que de fato representa uma seleção de cenários. Para o usuário final (o usuário que interage com o Home Assistant Frontend) a chamada de serviço de seleção de *playlist* e do dispositivo são apresentadas individualmente como um cenário, pois eles selecionam uma opção e ela é executada como se fosse setado o estado do componente com um valor padrão, mas no fundo é feito uma chamada de serviço com esse valor.

Levando em comparação todos os componentes, este *card* não representa um cenário de múltiplos componentes, pois é necessário escolher cenário a cenário e executar o *script* e ele seta cada cenário escolhido, ou seja, é setado o estado de cada dispositivo de IoT individualmente, não sendo selecionado o estado como um componente global.

Apesar de não ser a representação de um cenário de múltiplos componentes, ser um componente de múltiplos cenários tem suas vantagens, pois é possível selecionar cenário a cenário individualmente e executar todos com apenas um botão

A seleção de *playlist* e de dispositivos são a mesma representação do *card* de Spotify Playlist.

### 4.6.1 A seleção de cenário de iluminação

A seleção de cenário de iluminação funciona da seguinte maneira:

**O input:** O *input* de seleção representa a escolha entre 5 cenários de iluminação (Escolher Roupa, Festa, Romântico, Relaxante e Trabalhar de dia);

**O ícone:** Possui o ícone de uma lâmpada acesa;

**O data_template:** O data_template desta funcionalidade recebe o *input* como parâmetro e executa a seleção do cenário referente;

**O script:** O *script* faz a *call service* da seleção do cenário da mídia selecionando a Entity scene.turn_on.

### 4.6.2 Adicionando um novo serviço

Com a arquitetura detalhada, é possível que um novo serviço ou cenário possa ser utilizado em um cenário de múltiplos serviços, pois caso ele já exista e esteja escrito de acordo com a arquitetura definida, apenas é necessário adicionar sua chamada no *script*. Caso o serviço necessite de uma entrada de dados é necessário também adicioná-lo ao *group*.

#### 4.6.2.1 Adicionando uma cortina automatizada

Um exemplo de novo serviço que pode ser adicionado ao componente de Cenários Múltiplos seria o de iluminação artificial, fazendo o controle de uma cortina automatizada. Para isso seria necessário fazer a integração da cortina automatizada juntamente com o Home Assistant. Esta integração foi estudada, mas não houve tempo de ser implementada na solução final.

Para que esta solução fosse implementada seria necessário fazer com que o motor DC controlasse o movimento de abertura e fechamento da cortina e que a direção do movimento pudesse ser indicada através do relé Sonoff, fazendo assim sua automatização. Para que fosse possível a integração ao Home Assistant seria necessário trocar o software embarcado do Sonoff para o Tasmota, pois ele possui integração com Home Assistant caso o mesmo seja configurado para que o *broker* principal de comunicação de serviços seja o

MQTT, sendo necessário a utilização de um conversor de serial para USB para a inserção do novo sistema embarcado ao Sonoff.

Após estes passos, seria possível criar um serviço de controle da cortina através do Home Assistant e para que este serviço seja incluído a solução, sendo necessário seguir a arquitetura definida na solução e fazer sua chamada no *script* de Cenários Múltiplos.

#### 4.6.2.2 Desempenho do componente

A execução de cada funcionalidade do componente segue a seguinte ordem de execução e média de tempo:

**Seleção de cenário de iluminação:** 7,49 segundos;

**Seleção de dispositivo:** 10,12 segundos;

**Seleção de playlist:** 7,07 segundos.

Totalizando em média 24,68 segundos para a execução de todos os cenários.

## 4.7 Implementação do componente Cenários Completos

A implementação do *card* do componente de Cenários Completos resolve o problema apresentado na implementação do *card* Múltiplos Cenários 4.6, onde os mesmos cenários de iluminação e as *calls service* de seleção de *playlist* são utilizadas mas em vez de utilizar 2 *inputs* de seleção o mesmo *input* é utilizado para setar o estado deste cenário para todos os dispositivos de IoT envolvidos.

Para adicionar um novo serviço neste componente, funciona da mesma maneira do componente Múltiplos Cenários 4.6.

A execução de cada funcionalidade do componente segue a seguinte ordem de execução e média de tempo:

**Seleção de cenário de iluminação:** 7,77 segundos;

**Seleção de playlist:** 6,68 segundos.

Totalizando em média 14,45 segundos para a execução de todos os cenários.

## 4.8 Estudos complementares

Essa seção irá apresentar os estudos complementares realizados neste trabalho que foram feitos para melhorar o uso dos dispositivos utilizados e integração entre eles com outros elementos de IoT.

### 4.8.1 Implementações utilizando o IFTTT

O IFTTT foi uma tecnologia estudada e esta seção explora seu potencial para uma solução de automação residencial.

#### 4.8.1.1 Integração com Home Assistant

Primeiramente vale lembrar que o funcionamento do IFTTT se parece com o componente de *automation* do Home Assistant e é possível fazer com que os dois trabalhem em conjunto, fazendo com que uma ação dentro do Home Assistant seja disparada assim que um evento mapeado pelo IFTTT ocorra, ou que uma ação mapeada pelo Home Assistant dispare um evento pelo IFTTT. Assim é possível fazer com que uma casa seja uma aplicação dentro do IFTTT.

#### 4.8.1.2 Integrações feitas com IFTTT

Foram feitas implementações juntamente com IFTTT:

**Spotify e YouTube:** Quando o usuário der um *like* em uma música no YouTube se o Spotify tiver esta música, ela será adicionada a uma *playlist* específica;

**LIFX e Google Maps** Quando o usuário sair de uma área específica, todas as luzes serão apagadas;

**LIFX e Button Widget** Quando o usuário aciona o botão, um cenário é acionado. Este Button Widget pode ser utilizado com a tela bloqueada;

**Uber e LIFX** Quando o uber está chegando, a lampada Cama começa a piscar na cor verde.

Todas integrações feitas necessitaram de um gerenciamento de contas.

### 4.8.2 Implementações utilizando o HomeKit

Para facilitar o uso das lampadas LIFX pela Siri (assistente pessoal da Apple que aceita comandos de voz) foi feito a integração com o HomeKit. Para isso foi necessário apenas clicar em adicionar a acessório e escanear o QR Code pelo aplicativo do HomeKit.

Com isso é possível controlar as lampadas por comando de voz utilizando um *smart phone* da Apple. Para acionar comandos mais específicos como um cenário, primeiramente foi feito utilizado a ferramenta *shortcut* desenvolvida pela Apple, que permite que algum comando seja acionado ao por uma fala salva pela Siri. Uma solução mais eficaz feita, foi a importação dos cenários para o HomeKit possibilitando que eles sejam acionados pela Siri de forma mais rápida, além da fala não necessitar ser exatamente como a que foi gravada.

## 4.9 Contribuição com a comunidade do Home Assistant

Como comentado na seção 2.9.1.1, a arquitetura do Home Assistant envolve diversas camadas, sendo que este trabalho foi desenvolvido em cima da camada de Home Automation, onde foi possível criar alguns componentes personalizados em arquivos de configuração, ou seja, não foi submetido código para o código fonte do Home Assistant, pois não houve a necessidade.

Para que seja possível o compartilhamento de projetos assim como o apresentado neste trabalho, a comunidade do Home Assistant criou um canal de comunicação para facilitar a troca de conhecimento entre os membros. Este canal de comunicação possui diversas categorias entre elas o "Share your projects!"que possibilita que projetos sejam compartilhados.

Foi criado um novo tópico onde foi apresentado toda a solução deste trabalho, possibilitando o acesso a toda a comunidade do Home Assistant. O tópico está disponível no link:

<https://community.home-assistant.io/t/multiple-services-on-unique-scene/124722>

# 5 Conclusoes

O trabalho apresentado conseguiu chegar aos seus objetivos, havendo a implementação de um componente que possibilita a configuração de cenários utilizado múltiplos serviços selecionados por um usuário. Além do que foi um grande desafio a ser estudado e proporcionou grandes resultados.

Apesar dos problemas encontrados no decorrer da solução, o Home Assistant é uma ferramenta muito boa. Comparando com a outra ferramenta explorada o HomeKit, o Home Assistant possui mais componentes a serem integrados, sendo possível inclusive fazer a integração do HomeKit com o Home Assistant. Outra vantagem do Home Assistant em cima do HomeKit é a quantidade de customizações que é possível fazer, tais como as apresentadas na seção 2.9.1, principalmente quando é comparado as customizações que possibilita a criação de novos componentes.

Outro ponto a ser colocado em discussão é a integração do IFTTT com o Home Assistant, que possibilita com que uma casa seja ponte de integração com qual quer dispositivo que esteja conectado a Internet. Apesar de não ter sido necessário para a solução de um cenário mais completo utilizando o Home Assistant, penso que o IFTTT já abre as portas para que diversos serviços interaja com uma casa inteligente, e que com o passar do tempo haverá mais dispositivos que terão essa integração.

# 6 Trabalhos Futuros

Este capítulo representa os trabalhos futuros.

## 6.1 Ferramentas analisadas

Seção apresenta as ferramentas analisadas para futuras soluções.

### 6.1.1 Motor DC

Motor de corrente contínua que atuam aproveitando as forças de atração e repulsão geradas por eletroímãs e ímãs. Pode ser utilizado para controlar cortinas.

### 6.1.2 Sonoff

O relé *Wifi* sonoff funciona como um interruptor, com o diferencial de ser conectado a Internet por meio de um microcontrolador. Permitindo assim, conectar elementos como uma lâmpada comum a rede da casa. Ele trabalha com um regulador bivolt, podendo ser instalado tanto em redes elétricas de 110V quanto de 220V. Pode ser utilizado em conjunto ao motor DC para controlar a cortina.

### 6.1.3 Tasmota

*Software* embarcado alternativo que permite o controle do sonoff por meio do protocolo MQTT, para que ele possa ser integrado ao Home Assistant.

### 6.1.4 Conversor de serial para USB

Adaptador que converte uma entrada serial para USB. Necessário para que o Sonoff para atualizar com o Tasmota.

## 6.2 Problemas analisados

No decorrer do desenvolvimento da solução, foram encontrados 3 grandes problemas que dificulta a comercialização deste trabalho como produto e cada um deles gerou uma ideia de trabalho futuro.

O primeiro problema encontrado foi a seleção do dispositivo ChromeCast pelo Spotify através do Home Assistant, sem que este dispositivo já esteja conectado. Isso

acontece pois a biblioteca spotipy (biblioteca utilizada para interação com o Spotify pelo Home Assistant), ainda não consegue identificar o ChromeCast somente por ele se encontrar na mesma rede (como acontece pelo aplicativo mobile), sendo necessário para sua identificação, que ele esteja ligado. Por esta razão, não é possível selecionar uma *playlist* pelo ChromeCast sem que o aplicativo do Spotify esteja aberto, pois não tem como abrir o aplicativo do Spotify no ChromeCast através da interface do Home Assistant e para que isso aconteça é necessário fazer esta implementação em seu código fonte. Esta futura implementação irá agregar muito valor no produto final, pois com ela será possível através de um clique na interface do Home Assistant, disparar todos os eventos necessários para criar um cenário que envolva luz e música, até mesmo quando todos os dispositivos esteja inicialmente desligados.

O segundo problema encontrado é apagar as lampadas da LIFX sem o celular. Por a LIFX fazer sua automação através da lampada e não do interruptor, não é possível utiliza-la simultaneamente do interruptor, pois caso haja o uso do interruptor, a lampada fica sem energia. Uma alternativa para este problema seria substituir o interruptor comum por um botão e fazer com que este botão acione um cenário específico através do Home Assistant.

O terceiro problema encontrado é a complexidade da utilização Home Assistant por uma pessoa leiga. Apesar do Home Assistant ser um *hub* bastante completo, não é uma ferramenta muito intuitiva para ser utilizada por usuários que não tem base de programação. Por isso, acredita-se que para explora-lo comercialmente é necessário um modelo de negócios onde é feito toda instalação, configuração e manutenção da ferramenta.

Um outro trabalho futuro, não foi gerado através de um problema da solução, mas sim de uma possível evolução do produto. A adição de uma cortina automatizada ao componente Cenários Completos já foi explicada na seção 4.6.2.1.

# Referências

ABBOTT, M. Using music to promote l2 learning among adult learners. *Tesol*, v. 11, Dezembro 2011. Citado na página 30.

ARTEIRO, R. D. et al. Utilizando redes de petri para modelagem de desempenho de middleware orientado a mensagem. *WPerformance*, Julho 2007. Citado na página 34.

ASSOCIATION, G. Understanding the internet of things. *Connected Living*, Julho 2014. Citado na página 30.

BAUER, P. D. F. L.; BOLLIET, D. H. J. H. P. L. Software engineering. In: . [S.l.]: Peter Naur and Brian Randell, 1968. (NATO SCIENCE COMMITTEE). Citado 2 vezes nas páginas 32 e 33.

BIANCHI, G.; DAVID, P. L. D.; SUETA, R. U. Luz e cor nas unidades de hemodiálise: Estudo de caso da santa casa de misericórdia de araçatuba. *Revista Nacional de Gerenciamento de Cidades*, 2017. Citado na página 29.

BLUELUX. *O que é IFTTT?*: Você sabe como ele funciona. 2018. Online; accessed 24-Fevereiro-2018. Disponível em: <https://www.bluelux.com.br/o-que-e-ifttt-como-ele-funciona/?v=19d3326f3137>. Citado na página 36.

CARVALHO, H. *A Psicologia das Cores no Marketing e no Dia-a-Dia*. 2013. Online; accessed 04-Junho-2018. Disponível em: <https://viverdeblog.com/psicologia-das-cores/>. Citado 2 vezes nas páginas 25 e 29.

CATARINA, U. do Sul de S. Unisul de fato e de direito. *Revista Jurídica da Universidade do Sul de Santa Catarina*, v. 16, Janeiro a Julho 2018. Citado na página 25.

COMPUTERWORLD. *What is IFTTT?*: How to use if this, then that services. 2018. Online; accessed 24-Fevereiro-2018. Disponível em: <https://www.computerworld.com/article/3239304/mobile-wireless/what-is-ifttt-how-to-use-if-this-then-that-services.html>. Citado na página 36.

ERL, T. *SOA Principles of Service Design*. [S.l.]: DMCA, 2007. Citado na página 33.

FONSECA, J. F.; RHEINGANTZ, P. A. O ambiente está adequado? *Producao*, Dezembro 2009. Citado na página 29.

FREITAS, C. C. S. de et al. Automação residencial: Cenário atual e perspectivas futuras. *Revista Ciência e Tecnologia*, v. 15, Janeiro a Junho 2012. Citado na página 25.

GOUVEIA, M.; GOUVEIA, V. Service oriented architecture (soa): Desafios para o processo de desenvolvimento de software. *INESC*, 2004. Citado na página 33.

HOWE, D. *Free on-line dictionary of computing*. 2012. Online; accessed 19-Junho-2018. Disponível em: <https://web.archive.org/web/20120717095437/http://foldoc.org/>. Citado na página 32.

HUANG, R.-H.; SHIH, Y.-N. Effects of background music on concentration of workers. *Work*, v. 28, Dezembro 2011. Citado na página 30.

IBM. *O que é Middleware?* 2015. Online; accessed 19-Junho-2018. Disponível em: <https://www.ibm.com/middleware/br-pt/knowledge/>. Citado na página 32.

IBM. *Conhecendo o MQTT*: Por que o mqtt é um dos melhores protocolos de rede para a internet das coisas? 2018. Online; accessed 23-Fevereiro-2018. Disponível em: <https://www.ibm.com/developerworks/br/library/iot-mqtt-why-good-for-iot/index.html>. Citado na página 35.

IQBAL, M. *Spotify Usage and Revenue Statistics*. 2019. Online; accessed 2-Julho-2019. Disponível em: <https://www.businessofapps.com/data/spotify-statistics/>. Citado na página 47.

KALE, V. *Cloud computing for business and technology managers*: from distrbuted computing to cloudware applications. [S.l.]: CRC Press, 2015. v. 1. Citado na página 34.

KOTH, D. A influência da iluminação e das cores no ambiente hospitalar: a saúde vista com outros olhos. *Especialize*, Janeiro 2013. Citado na página 29.

MANAIA, M. B. A influência da iluminação no comportamento humano. *IJEP Instituto Junguiano*, 2013. Citado na página 29.

MAS, I.; RADCLIFFE, D. Mobile go viral: M-pesa in kenya. *Bill & Melinda Gates Foundation*, Março 2010. Citado na página 31.

MASSE, M. *REST API Design Rulebook*: Designing consistent restful web service interfaces. [S.l.]: O'Reilly Media, 2012. v. 1. Citado na página 34.

ORG, M. *Frequently Asked Questions*. 2014. Online; accessed 23-Fevereiro-2018. Disponível em: <http://mqtt.org/faq>. Citado na página 35.

OVERY, K.; SZAKAC, I. M. Being together in time: Musical experience and the mirror neuron system. *Digital Publishing*, v. 26, Junho 2009. Citado na página 29.

PARNCUTT, R.; MCPHERSON, G. E. *The science and psycology of music perfomace: Creative strategy for teaching and learning*. [S.l.]: Oxford, 2002. v. 1. Citado na página 30.

PRODANOV, C. C.; FREITAS, E. C. de. *Metodologia do trabalho cinetífico*: Métodos e técnicas da pesquisa e do trabalho acadêmico. [S.l.]: Feevale, 2013. v. 2. Citado na página 43.

ROCHA, V. C. da; BOGGIO, P. S. A música por uma óptica neurocientífica. *Per Musi*, Junho 2013. Citado 3 vezes nas páginas 25, 29 e 30.

ROUSE, M. *Smart home hub (home automation hub)*. 2018. Online; accessed 24-Fevereiro-2018. Disponível em: <https://internetofthingsagenda.techtarget.com/definition/smart-home-hub-home-automation-hub>. Citado na página 36.

STANDLEY, J. M. Does music instruction help children learn to read? evidence of a meta-analysis. *Sage Pub*, Novembro 2008. Citado na página 30.

TANENBAUM, A. S.; STEEN, M. V. *Distributed systems: Principes and Paradigms*. [S.l.]: Pearson Education Inc, 2006. v. 2. Citado na página 31.

VALNET, C. *Chromotherapy the power of colors*. [S.l.]: Edizioni, 2015. v. 1. Citado na página 29.

VERMESAN, O.; FRIESS, P. *Internet of Things - Converging Technologies for Smart Environments and Integrated Ecosystems*. [S.l.]: River Publisher, 2013. v. 1. Citado na página 31.

# Apêndices

# APÊNDICE A – Instalação do Home Assistant

Em relação à evolução do Home Assistant, primeiro foi necessário instalar um sistema operacional na Raspberry Pi e para isso foi escolhido o Raspbian. Uma alternativa seria instalar o Hassbian, um sistema operacional que possui o Home Assistant embutido em sua imagem. Ele não foi escolhido pois, para uma possível mudança no código fonte, seria necessário gerar uma nova imagem do sistema operacional.

Para instalar a imagem Raspbian, foi necessário baixa-la em um computador, escrever a imagem em um cartão SD e inseri-lo na entrada da Raspberry, para então fazer a inicialização do sistema.

A instalação do Home Assistant foi feita pelo pacote *pip* "homeassistant". Para o sucesso dessa solução, foi necessário instalar uma versão do Python igual ou superior que a 3.5.3 (versões que o Home Assistant é suportado), sendo que a versão escolhida foi a do Python 3.6. Foi necessário baixar o código fonte do Python, e compila-lo, pois a versão mais atualizada do Python que o Raspbian possui empacotado até o momento é a 3.4.

Para isolar a instalação dos pacotes Python no ambiente de desenvolvimento, foi utilizado o Virtualenv. Para acessar o ambiente virtual é necessário entrar dentro da pasta homeassistant/ e rodar o comando:

```
source bin/activate
```

Após o levantamento do servidor através da seguinte linha de comando:

```
hass --open-ui
```

O acesso da interface do Home Assistant no computador pode ser feita através do endereço:

```
http://hassio:8123/
```

Isso foi possível pois o hassio permite que qualquer dispositivo dentro da mesma sub rede, tenha acesso a sua interface, após o levantamento do servidor.

Após a estabilidade dos componentes desenvolvidos no projeto em questão, a Raspberry pôde de fato funcionar como um Hub, pois com a junção de algumas funcionalidades,

começou a fazer sentido ter um app que centraliza o controle de diversos dispositivos. Para que a Raspberry possa funcionar como Hub é necessário levantar o servidor do hassio como *deamon*, pois assim, não há necessidade de toda vez em que o algum usuário do Home Assistant sinta necessidade de seu uso, seja necessário rodar o servidor, ou seja, ligar um computador que tenha toda a configuração para acesso a Raspberry, fazer a conexão via SSH, entrar na pasta homeassistant/ ativar o Virtualenv, e levantar o servidor do hassio. Isso acontece pois com o servidor rodando como *deamon* sua execução é feita em *background*, logo, só há necessidade da Raspberry ficar ligada e ela estará funcionando de fato como um Hub. Para levantar o servidor do hassio como *deamon*, é necessário a seguinte linha de comando:

```
hass --open-ui --deamon
```

Com a Raspberry funcionando como *deamon*, foi possível utilizar as versões mobile do Home Assistant. Para isso foi necessário baixar o app na PlayStore para os dispositivos Android, e na AppStore para dispositvos IOS. Em alguns casos com o uso do Android foi necessario utilizar a versão PWA, pois não foi encontrado o app da forma nativa.

# APÊNDICE B – Configuração da Raspberry Pi e rede

Na implementação do projeto, havia uma limitação de recursos para acessar a Raspberry, não havendo nem teclado nem monitor, mas havia um cabo *Ethernet*. Desta maneira, foi necessário criar uma rede cabeada IPv4 de compartilhamento de rede e fazer com que a Raspberry permita-se ser acessada via SSH. Ao criar uma nova conexão IPv4 a Raspberry pôde ser acessada via SSH pelo IP que aparaceu no retorno do comando:

```
cat /var/lib/misc/dnsmasq.leases
```

Desta forma o comando de acesso a Raspberry via SSH foi:

```
ssh pi@IP
```

Ao executar esse comando, é necessário digitar a senha do usuário "pi"da Raspberry. Para facilitar o processo e não ser mais necessário inserir a senha toda que for acessar a Raspberry via SSH, foi gerado uma chave no computador pessoal com o comando:

```
ssh-keygen
```

Esse comando utiliza um algorítimo de chave pública e privada e cria duas chaves (publica e privada) dentro da pasta "~/.ssh/". A chave publica fica armazenada no arquivo "~/.ssh/id_rsa.pub"do computador pessoal e deve ser copiada para o arquivo "~/.ssh/authorized_keys"na Raspberry, desta forma não é necessário inserir a senha ao acessar a Raspberry via SSH a partir do computador pessoal.

O passo seguinte foi configurar o WiFi na Raspberry, tanto para não utilizar mais o cabo *Ethernet*, quanto para o servidor Home Assistant conseguir identificar os dispositivos na rede. Para isso foi necessário colocar as seguintes configurações no arquivo "/etc/wpa_supplicant/wpa_supplicant.conf":

```
network={
  ssid="Nome Da Rede"
  psk="Senha da Rede"
}
```

Também foi necessário setar um IP estático no arquivo "/etc/network/interfaces", pois algumas configurações do Home Assistant necessitam que o IP não mude, além de facilitar o acesso tanto ao servidor que provê a interface gráfica do Home Assistant, quanto a Raspberry. A seguir está apresentado a configuração com os IPs reais da solução:

```
allow-hotplug wlan0
iface wlan0 inet static
        address 192.168.0.17
        netmask 255.255.255.0
        gateway 192.168.0.1
    wpa-conf /etc/wpa_supplicant/wpa_supplicant.conf
```

Ao definir um IP estático na Raspberry foi possível configurar o arquivo "~/.ssh/config"no computador pessoal da seguinte maneira:

```
Host pi
  HostName 192.168.0.17
  User pi
```

Desta forma, o acesso a Raspberry pelo computador pessoal via SSH se reduziu ao seguinte comando:

```
ssh pi
```

Não sendo necessário assim, nem indicar o IP da máquina, nem o usuário (apesar de coincidir o usuário com o nome que foi definido para realizar o acesso).

A última configuração de ambiente feita foi a do arquivo "/etc/hosts":

```
192.168.0.17  hassio
```

Essa configuração permitiu que o endereço 192.168.0.17 da Raspberry fosse mapeado com o nome hassio. Esse arquivo pode ser comparado a um serviço de resolução de nomes, em um escopo local.

# APÊNDICE C – Fotos dos cenários de iluminação

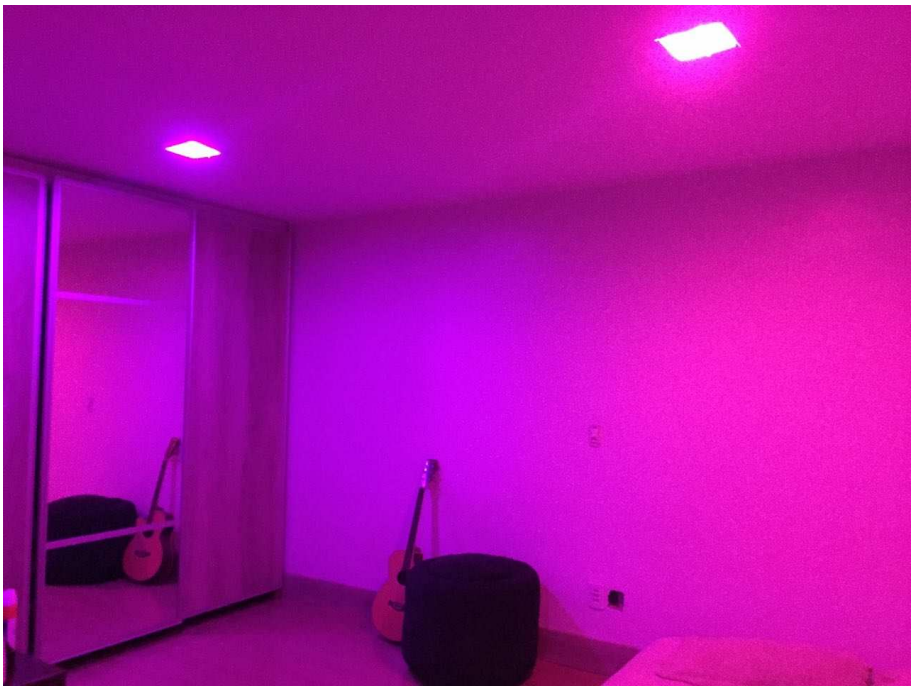

Figura 15 – Cenário Festa

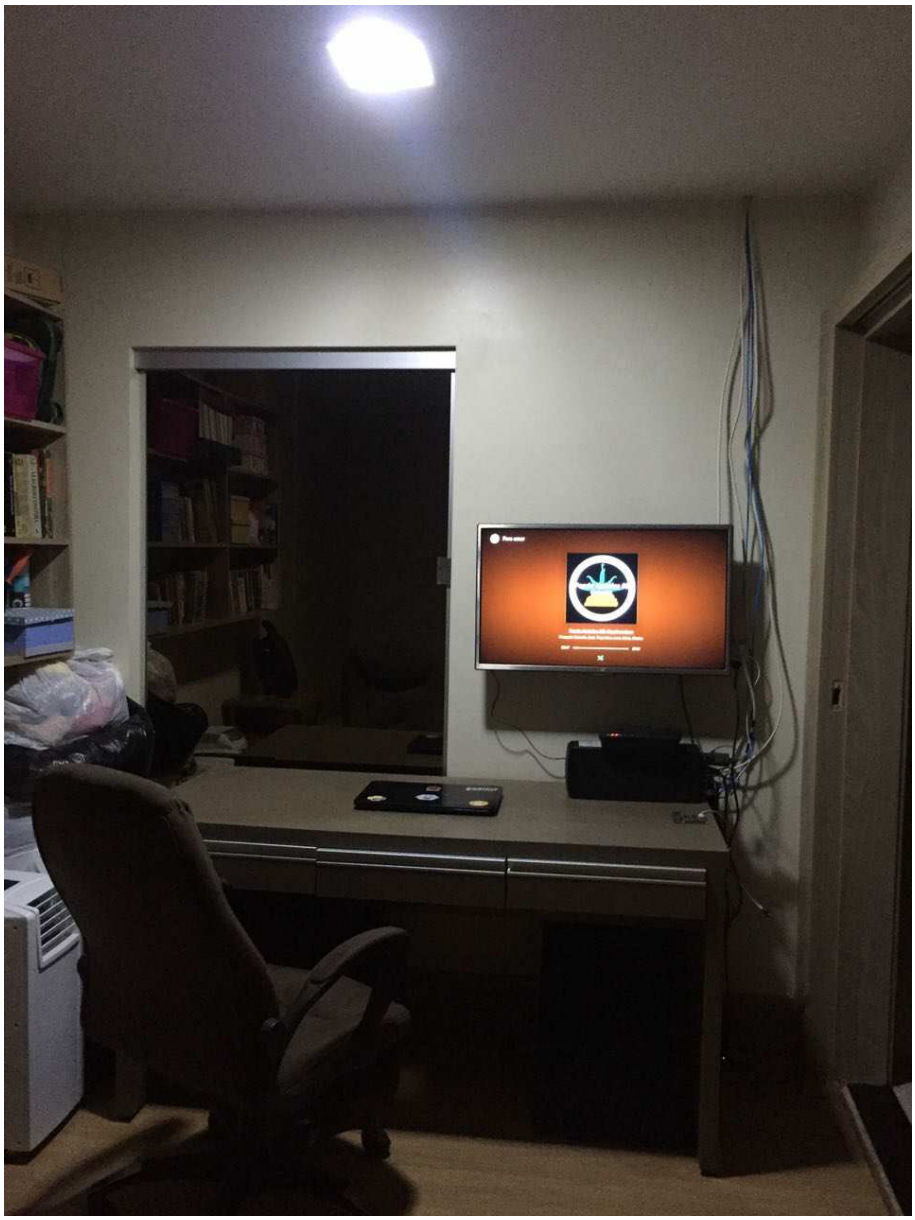

Figura 16 – Cenario trabalhar de dia

Figura 17 – Cenario escolher roupa