UNIVERSIDADE FEDERAL DE SANTA MARIA
PROGRAMA DE CIÊNCIA DA COMPUTAÇÃO

Rafael Pavani Guimarães

ARQUITETURA E ESTUDO DE VIABILIDADE DE SISTEMA CIENTE DE LOCALIZAÇÃO PARA CONTROLE DE DISPOSITIVOS

Santa Maria, RS
2021

Rafael Pavani Guimarães

ARQUITETURA E ESTUDO DE VIABILIDADE DE SISTEMA CIENTE DE LOCALIZAÇÃO PARA CONTROLE DE DISPOSITIVOS

Trabalho de Conclusão de Curso apresentado ao Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de **Bacharel em Ciência da Computação**

Orientador: Prof. Dr. Celio Trois

494
Santa Maria, RS
2021

Pavani Guimarães, Rafael

Arquitetura e Estudo de Viabilidade de Sistema Ciente de Localização para Controle de Dispositivos / por Rafael Pavani Guimarães. – 2021.

3 f.: il.; 30 cm.

Orientador: Celio Trois

Trabalho de Conclusão de Curso - Universidade Federal de Santa Maria, , Ciência da Computação, RS, 2021.

1. IoT. 2. Ambientes Inteligentes. 3. API. 4. Arquitetura. I. Trois, Celio. II.Arquitetura e Estudo de Viabilidade de Sistema Ciente de Localização para Controle de Dispositivos.

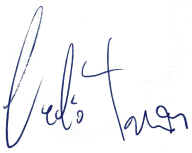
E-mail: rpguimaraes@inf.ufsm.br

Rafael P. Guimarães

ARQUITETURA E ESTUDO DE VIABILIDADE DE SISTEMA CIENTE DE LOCALIZAÇÃO PARA CONTROLE DE DISPOSITIVOS

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação, da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 31 de Agosto de 2021



Celio Trois, Dr. (UFSM)
(Presidente/Orientador)

João Carlos Damasceno Lima, Dr. (UFSM)

Antonio Marcos De Oliveira Candia, Me. (UFSM)

Santa Maria, RS
2021

AGRADECIMENTOS

Agradeço ao meu orientador Celio Trois por conduzir o meu trabalho com excelência.

Agradeço aos professores do curso de Ciência da Computação da UFSM pela elevada qualidade nos ensinamentos a mim passados.

Agradeço aos meus pais e à minha família por todo o apoio durante o meu trajeto neste curso e na minha vida.

Arquitetura e Estudo de Viabilidade de Sistema Ciente de Localização para Controle de Dispositivos

Rafael Pavani Guimarães

[1]Centro de Tecnologia – Universidade Federal de Santa Maria (UFSM)

rpguimaraes@inf.ufsm.br

Abstract. *Recent inovations in IT field have been heavily focusing the Internet of Things (IoT), revolutionizing the domestic ambients. It is possible to notice an increasing amount of smart objects, such as intelligent appliances, TVs, Smartphones, and various types of sensors being conected through a network. With this revolution, a large amount of possibilities emerges for daily activities automation, granting more free time and confort to users. This work proposes the architecture of a system that automates the activation and deactivation of devices that are part of a domestic network, so that only the devices nearest and most relevant to the user are utilized. A study about the viability was made about tools that can control devices; more specifically: the Spotify API, the Google Assistant gRPC API, the Google Assistant Relay and the Fulfillment. It was determined that the Spotify API, the Google Assistant gRPC API and the Google Assistant Relay do not fulfill the requirements for the proposed system, as they cannot control the audio devices. The study made about the Fulfillment demonstrated that it may fulfill the requirements, however, because of a lack of time, it was not possible to perform the necessary tests to prove if the tool fulfilled the requirements of the proposed architecture.*

Resumo. *Recentes inovações na área de TI vêm sendo fortemente focadas em Internet das Coisas (IoT), revolucionando os ambientes domésticos. É possível perceber cada vez mais a existência de objetos inteligentes que estão conectados em rede, como eletrodomésticos inteligentes, TVs, smartfones e diversos tipos de sensores conectados por uma rede. Com esta revolução, várias possibilidades surgem para a automatização de atividades do cotidiano, liberando tempo livre e aumentando o conforto para usuários. Este trabalho propõe a arquitetura de um sistema que automatiza o ligamento e desligamento de dispositivos que façam parte de uma rede doméstica, de forma que sempre os dispositivos mais próximo e relevantes ao usuário sejam utilizados. Um estudo de viabilidade foi feito sobre ferramentas existentes que permitem o controle de dispositivos; mais especificamente Spotify API, Google Assistant gRPC API, Google Assistant Relay e Google Fulfillment. O estudo demonstrou que o Spotify API, o Google Assistant gRPC API e o Google Assistant Relay não atendem os requisitos básicos do sistema proposto, pois não permitem o controle dos dispositivos de áudio. O estudo feito com o Fulfillment demonstrou ser uma alternativa que atende os requisitos, porém, por falta de tempo, não foi possível efetuar os testes necessários para comprovar sua utilização na arquitetura proposta.*

1. Introdução

Nos últimos anos, o avanço da tecnologia tem sido cada vez mais veloz e tem provocado uma mudança nos interesses e preferências de usuários por serviços sensíveis ao contexto, situacionais e personalizados. Dentro da área de Internet das Coisas (*Internet of Things* - IoT), área esta que tem se desenvolvido muito nos últimos anos [Alaa et al. 2017], existem vários sistemas de Ambientes Inteligentes que proporcionam serviços mais personalizados e que resolvem problemas específicos. Estas soluções podem ser aplicadas em conjunto com várias outras áreas do conhecimento, como por exemplo, na área da saúde ajudando na monitoração de um idoso dentro de sua casa [Chen et al. 2015], ou na área de Engenharia Civil, para reduzir o consumo de energia elétrica dentro de uma casa [Yin et al. 2015].

Na medida que as áreas de IoT e Ambientes Inteligentes se desenvolvem, a infraestrutura evolui, integrando a mais objetos do cotidiano sensores e computadores e com cada vez mais funcionalidades. Em certos casos, múltiplos dispositivos inteligentes coexistem em um ambiente, como uma casa, que pode ter um ou mais dispositivos em cada cômodo, funcionando de forma independente [Al-Kuwari et al. 2018].

Considerando a evolução na infraestrutura e nos ambientes inteligentes, surgem novas possibilidades de sistemas de automação de atividades rotineiras, com o intuito de melhorar a qualidade de vida de usuários [Kodali et al. 2016]. Visto que existem cada vez mais dispositivos eletrônicos dentro das casas, problemas surgem referentes à tarefa de desligar dispositivos que não irão mais ser usados, e também à de ligar dispositivos que irão ser usados.

Este trabalho propõe a arquitetura de um sistema que utiliza um controlador, dispositivos que disponibilizem serviços ao usuário e sensores capazes de informar a localização do usuário, conectados através de uma rede local. Este ambiente é então capaz de utilizar a localização do usuário, de forma que apenas os dispositivos que apresentem um serviço relevante para o usuário, como um ar-condicionado, sejam ativados e que dispositivos que não sejam relevantes para o usuário desativem os seus serviços. Sempre que algum sensor captar um movimento, ele enviará um sinal para o controlador, para que este então, sabendo da posição do usuário, possa enviar as ordens de ativação e desativação dos serviços para os dispositivos apropriados.

Após definida a arquitetura, foi feito um estudo de viabilidade nas ferramentas necessárias que possibilitam a implementação do sistema proposto na arquitetura. Foram estudadas e testadas uma API do *Spotify* e as outras três do *Google*, respectivamente *Spotify API*, *Google Assistant gRPC API*, *Google Assistant Relay* e *Fulfillment*. Após estudar e testar as três primeiras abordagens, descobriu-se que todas apresentavam limitações que impediam a execução desta proposta. Na sequência deu-se início ao estudo da quarta opção (*Fulfillment*) e, por falta de tempo, não pode ser implantada e testada.

Este trabalho será organizado da seguinte forma: A Seção 2 relata trabalhos relacionados constantes na bibliografia atual. A Seção 3 apresenta a arquitetura de forma abstrata do sistema final, juntamente com um exemplo demonstrando seu funcionamento. A Seção 4 apresenta o estudo e viabilidade das APIs testadas para a implementação da parte de controle de dispositivos do sistema e por fim a Seção 5 conclui o presente trabalho.

2. Trabalhos Relacionados

Nesta seção são apresentados trabalhos relacionados com a proposta. Primeiramente são apresentados trabalhos que facilitam a interação do usuário com o ambiente e, em seguida, propostas que usam a localização do usuário para automatizar certas tarefas.

O trabalho de [Mainetti et al. 2015] propõem um eco-sistema de software que pessoas com diferentes níveis de habilidade possam configurar. Neste software o usuário define regras de serviços, os quais o sistema irá usar em conjunto com a localização do usuário para fazer o controle do ambiente. [Yachir et al. 2015] propõem um Framework centrado no usuário como um sistema para controle de eventos e entidades do ambiente abstraídas como serviços de ambiente e faz um estudo sobre a interação entre os diversos dispositivos conectados em rede e o Framework. Este controle é feito através de três principais tipos de serviços de ambientes, nomeadamente, serviços sensíveis a eventos, serviços de controle de dispositivos e serviços sensíveis ao contexto.

O trabalho de [Moreno et al. 2014] propõem um exemplo de um sistema IoT centrado no usuário para conseguir reduzir o consumo de energia em um edifício através da consideração dos comportamentos dos usuários para a gestão da infraestrutura do edifício. Este sistema experimental foi então testado em escritórios, comprovando ser possível reduzir o consumo de energia. [Alletto et al. 2015] desenvolveu e validou uma arquitetura ciente da localização para ambientes internos que, no contexto do trabalho, melhoram a experiência de usuários dentro de um museu. Este sistema faz a localização através do uso de sensores Bluetooth de baixo consumo e um dispositivo que o usuário tem vestido. Através da localização, outros serviços podem ser oferecidos aos usuários.

Considerando os estudos apresentados nesta seção, podemos considerar que o conhecimento sobre a localização e monitoração de usuários é aparente tanto no trabalho de [Alletto et al. 2015] quanto no trabalho de [Mainetti et al. 2015]. Quanto à forma de como são tratados os contextos e ambientes neste sistema é levado em conta o trabalho de [Moreno et al. 2014] que demonstra formas de enxergar os usuários em certos contextos e o [Yachir et al. 2015] que explica e demonstra estes detalhes também no formato de serviços de ambiente. Esses conceitos são usados na arquitetura proposta neste trabalho, a qual será detalhada na próxima seção.

3. Arquitetura do Sistema

Este trabalho propõem um sistema que utiliza a localização do usuário para controlar dispositivos. A localização do usuário se dá pelo meio de sensores espalhados pelo ambiente conectados a uma rede local. Juntamente com os sensores, a rede também incorpora dispositivos inteligentes que possuam funcionalidades relevantes para o usuário e um controlador. Com a localização passada pelos sensores, o controlador é capaz de ativar os serviços de dispositivos relevantes e desativar serviços de dispositivos que não sejam mais necessários.

Os trabalhos relacionados promoveram o uso de sensores de bluetooth devido à sua precisão e facilidade de identificação do usuário [Alletto et al. 2015], onde os sensores localizam e identificam o usuário através do seu celular. Neste tipo de sensor, é importante lembrar que é necessário que o usuário sempre tenha em sua posse o seu celular. Apesar deste problema, foi ainda considerado para a arquitetura do sistema os sensores bluetooth,

por conta das vantagens quanto à maior simplicidade na localização e identificação do usuário [Mackensen et al. 2012] e também visto que considerar que o usuário sempre está com o seu celular dentro do ambiente não é algo inconcebível.

A arquitetura é composta também por um servidor local principal que existe dentro do ambiente onde a lógica do sistema reside, sendo este servidor um hardware com capacidade de manter uma rede local para os dispositivos para além do sistema. O controlador pode ser, por exemplo, a *Amazon Alexa*, que conecta os dispositivos inteligentes e permite o controle destes através de comandos de voz.

Os tipos de dispositivos que o sistema proposto abrange são qualquer dispositivo que possua um serviço que seja relevante para o usuário de acordo com a sua posição. Dispositivos estes podendo ser, por exemplo, um ar-condicionado, cujo serviço de controle de temperatura de um cômodo apenas é relevante se o usuário estiver no mesmo cômodo. Outro exemplo é o da reprodução de algum tipo de mídia, como por exemplo música, onde não é relevante que esta reprodução ocorra longe do usuário visto que o mesmo não irá conseguir escutar o que está sendo reproduzido.

Na Figura 1 é feita a especificação de forma visual de como a arquitetura foi estruturada. À esquerda, são representados os dispositivos e sensores físicos juntamente com os dados principais usados na comunicação para com o servidor controlador, e à direita o servidor controlador onde o sistema final reside. O servidor recebe as leituras dos sensores com informações sobre a localização e a identificação do usuário, identificando quais os dispositivos e serviços são relevantes através de uma tabela que faz o mapeamento dos dispositivos e os serviços que cada um dispõe, levando em conta que os dispositivos que disponibilizem serviços sejam estáticos dentro do ambiente. Dentro do controlador existe um coordenador de ações, que determina quais as ações a serem tomadas para os serviços passados pela tabela e, caso existam, os serviços guardados que estão em uso e que devem sofrer uma ação. O coordenador então aplica as ações através da API de controle.

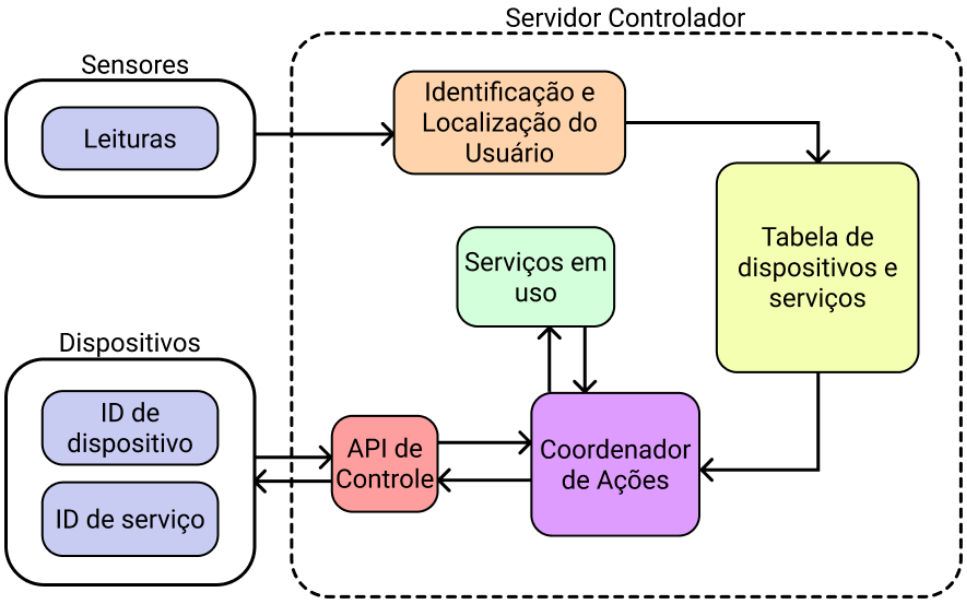


Figura 1. Arquitetura do sistema

Quando os sensores fazem uma leitura da localização do usuário, esta leitura é

passada para o servidor, que identifica em quais dispositivos irá agir através da tabela de associação entre dispositivos e localizações. Após identificados os dispositivos onde serão executadas ações, o servidor controlador decide então quais são as ações que serão aplicadas em cada dispositivo, ações estas estando definidas em um coordenador de ações. Para o controlador aplicar as ações em um dado dispositivo é utilizado uma API de controle de dispositivos.

Na Figura 2 é demonstrado um exemplo de funcionamento do sistema. Neste exemplo um usuário que está dentro de sua casa se locomove do seu escritório, que possui um computador que está reproduzido um serviço, como um vídeo, para a sala, que possui uma *smart-TV* em *standby* (passo 1). Ao entrar na sala os sensores presentes neste cômodo são ativados (passo 2) e passam um sinal para o controlador contendo a nova posição do usuário (passo 3). Em seguida, o controlador executa as seguintes funções em paralelo (passo 4): para a *smart-TV*, que está na sala, é passada a função de retomar a reprodução do vídeo que originalmente estava sendo reproduzido no computador do escritório e para esse computador que está no escritório é passada a função de parar de reproduzir o vídeo.

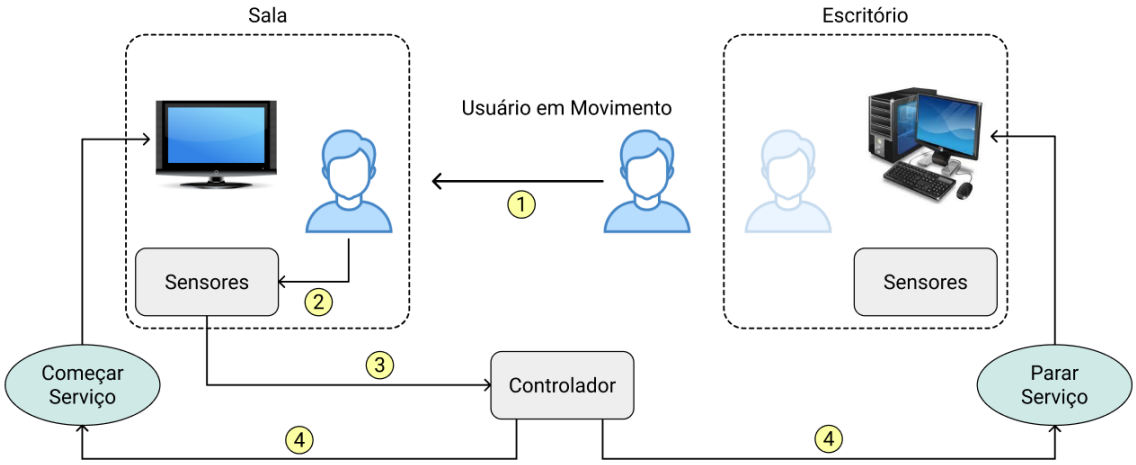


Figura 2. Exemplo do fluxo de trabalho do sistema

Na próxima seção são apresentadas as APIs e ferramentas que foram estudadas no âmbito de possibilitar a criação do sistema proposto na arquitetura.

4. Estudo de Viabilidade

Para produzir um estudo mais tangível em relação a ferramentas de controle de dispositivos, como caso de uso foram estudadas ferramentas que conseguissem, em particular, controlar a reprodução de áudio em dispositivos. Para isso, foram realizadas buscas por possíveis abordagens que pudessem fazer esse controle, onde, no decorrer da pesquisa, foram encontradas quatro possibilidades que serão apresentadas no decorrer desta seção. Primeiramente foi estudada a *Spotify API* e, em seguida, foram estudadas três possíveis ferramentas da *Google*, sendo elas: o *Google Assistant gRPC API*, o *Google Assistant Relay* e o *Fulfillment*.

4.1. *Spotify API*

Primeiramente estudou-se a API que permite controlar o aplicativo *Spotify*[1]. Foram testadas as chamadas da API que permitem transferir a reprodução do áudio de um dispositivo para outro. Mais especificamente as chamadas que foram testadas foram: *Get Device*[2], que realiza uma busca de dispositivos considerados ativos dentro do *Spotify* e *Transfer User Playback*[3], que executa uma mudança de reprodução da música de um dispositivo para outro.

A funcionalidade *Get Device* da API faz uma busca por todos os dispositivos que estão com o aplicativo *Spotify* aberto para uma conta específica e então retorna um arquivo no formato `JSON` contendo informações (ex.: ID, nome, status) sobre os dispositivos encontrados pela API.

A funcionalidade *Transfer User Playback* tem como parâmetro um ID de dispositivo que é o dispositivo alvo para qual vai ser feita a transferência da reprodução de áudio e, ao executar a funcionalidade, o dispositivo que está no momento reproduzindo áudio interrompe sua reprodução, e o dispositivo alvo retoma a reprodução do ponto onde o primeiro dispositivo parou.

Para testar o funcionamento desta API foi criado um servidor `localhost` utilizando Node.js e um código exemplo proporcionado pela *Spotify for Developers*, onde neste servidor local é possível fazer chamadas à API.

Após testes do código exemplo e estudos mais profundados da documentação foi percebido um problema em relação a esta API. Apenas dispositivos onde o aplicativo *Spotify* estiver aberto são reconhecidos pela API. Esta API também não tem a capacidade de abrir o aplicativo em um dispositivo. Por conta destes problemas que vão contra a ideia de automação proposta neste trabalho, foi descartada esta abordagem para o controle de dispositivos.

4.2. *Google Assistant SDK*

Devido às limitações da API do *Spotify*, iniciou-se a busca por outras ferramentas de controle de dispositivos. Uma segunda ferramenta foi identificada: a *Google Assistant SDK*[4], que apresentou permitir um controle de dispositivos de forma mais abrangente em relação à integração do sistema com dispositivos externos através da *Google Assistant*.

A *Google Assistant SDK*[5] inicialmente apresentou problemas, visto que não existe o aplicativo *Google Assistant* nativo para Windows ou para Linux. Ao longo do estudo desta ferramenta foi percebido que ela possui duas funções principais: a criação de uma ação e o envio dessa ação para o *Google Assistant* para ser tratada. A criação de uma ação pelo SDK envolve o reconhecimento de um comando de voz ou de texto e construção de uma ação formalizada com base nesse comando, de forma que o *Google Assistant* consiga reconhecer a ação e executá-la. Mais especificamente, o SDK reconhece a entrada

[1]Spotify: `https://developer.spotify.com/`

[2]Get Device: `https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-a-users-available-devices`

[3]Transfer User Playback: `https://developer.spotify.com/documentation/web-api/reference/#category-player`

[4]https://developers.google.com/assistant

[5]Disponível em: `https://developers.google.com/assistant/sdk/overview`

e o transforma em um JSON contendo as informações de entrada; este é enviado para a *Google Assistant*.

Para fazer o uso das APIs da *Google* foi necessário primeiro a criação de um projeto no *Google Actions*, ferramenta online para criação de ações que a *Google Assistant* pode executar e que incorporem diretamente as outras ferramentas e APIs da *Google*, funcionando juntamente com o *Google Cloud Plataform* como conector entre *front-end* e *back-end* de projetos. Para o funcionamento correto do *Google Actions* e do *Google Cloud Plataform*, é necessário fazer um controle de credenciais chamado `OAuth 2.0`, onde é gerado para o cliente um ID público e um ID secreto que têm de ser incorporados por outros sistemas que estão acessando o *Google Actions* para possibilitar o uso deste e do *Google Cloud Plataform*.

Após a criação do projeto no *Google Actions*, foram estudadas três ferramentas que conseguissem controlar dispositivos com base na *Google Assistant API*, sendo elas: *Google Assitant gRPC API*, *Google Assistant Relay* e o *Fulfillment*, descritas nas próximas subseções.

4.2.1. *Google Assistant gRPC API*

O gRPC[6] é um framework para a comunicação entre sistemas e serviços criado pela *Google* que serve como alternativa à API REST. O gRPC pode ser usado para acessar a API do *Google Assistant*.

Análises iniciais apontavam que esta ferramenta possuía as mesmas funcionalidades que o aplicativo da *Google Assistant* presentes em sistemas operacionais *Android*. Porém, os testes e estudos feitos com essa API apresentaram limitações quanto à interação com aplicativos terceiros.

Foi implementado um código de teste baseado no exemplo da *Google Assistant gRPC API*[7], onde foi possível constatar que a API é capaz de executar requisições de ações para pesquisas simples no motor de busca *Google*. Porém, dentre os testes realizados, a API demonstrou-se incapaz de abrir ou interagir com outras aplicações. Devido ao insucesso de interagir com outros aplicativos, esta API foi descartada para o sistema proposto, iniciando-se uma nova procura por uma ferramenta para o controle de dispositivos.

4.2.2. *Google Assistant Relay*

O *Google Assistant Relay*[8] é uma ferramenta que permite a delegação de uma ação executável pelo aplicativo *Google Assistant* para um dispositivo. A ferramenta expõe o *Google Assistant* como uma REST API funcionando sobre o `Node.js` e usando o *Google SDK* para comunicação com os serviços da *Google*.

Testes foram feitos na versão 3.2.0 da ferramenta e demonstraram que a esta não era capaz de executar uma invocação de multimídia visto que esta funcionalidade ainda

[6]Disponível em: `https://grpc.io/`

[7]Disponível em: `https://github.com/googlesamples/assistant-sdk-python/tree/master/google-assistant-sdk/googlesamples/assistant/grpc`

[8]Disponível em: `https://assistantrelay.com/`

não foi implementada e era apenas capaz de realizar invocações mais simples, como a de executar uma pesquisa no motor de busca *Google*. Segundo a documentação da ferramenta quanto à invocação de multimídia[9]: "*This feature is currently a Work In Progress, and is only available in v3.3b at this time*", onde a versão 3.3b se trata de uma versão beta.

Visto que a ferramenta, no período deste trabalho, não possui a capacidade de executar a invocação de multimídia, ela foi descartada como possibilidade para com o sistema proposto.

4.3. *Fulfillment*

Após as tentativas com as duas ferramentas da *Google* de front-end não serem bem sucedidas, o estudo se voltou para uma ferramenta de back-end que se integra ao *Google Actions* e ao *Google Cloud*. O *Google Actions* tem a capacidade de delegar lógica de serviços para *HTTPS Web services*, denominados *Fulfillment*[10], através de um `Webhook`[11] que faz uma requisição ao *Fulfillment* para criar uma resposta, enviando um *Payload* anexado à requisição.

Dentro de um *Fulfillment* existem `Handlers` de `Webhooks` que fazem o processamento do *Payload* de acordo com o tipo de requisição para criar a resposta esperada para aquele `Webhooks`.

Durante os estudos sobre a implementação de um *Fulfillment* foram descobertos os serviços *Firebase* que são de propriedade da *Google* e que proporcionam suporte ao desenvolvimento de aplicações, sendo mais relevantes para este trabalho a criação e hospedagem de um *Fulfillment*.

Para a criação de um *Fulfillment* o *Firebase* oferece o *Cloud Functions for Firebase*[12], um framework que permite a execução de código em resposta a eventos, mais especificamente para este caso a requisição através de `Webhooks`.

Considerando que foi necessário um tempo não previsto na proposta para testar essas APIs, não foi possível fazer a construção e testes com o *Fulfillment* para averiguar se esta ferramenta atenderia aos requisitos da proposta descrita na arquitetura.

[9]Casting Media: https://assistantrelay.com/docs/cast/casting/

[10]Fulfillment: `https://developers.google.com/assistant/smarthome/concepts/fulfillment-authentication/`

[11]Webhook: https://developers.google.com/assistant/conversational/webhooks

[12]Cloud Functions for Firebase: https://firebase.google.com/docs/functions

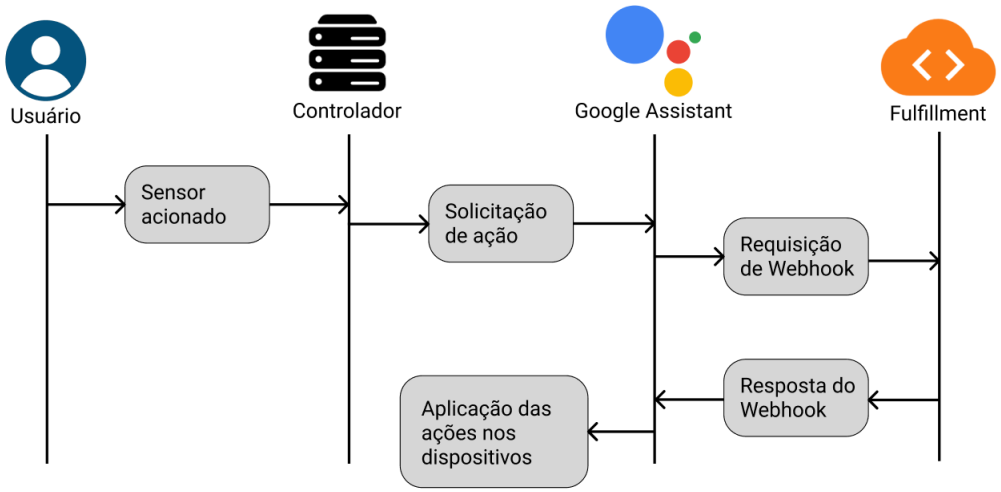


Figura 3. Funcionamento de um Fulfillment no sistema proposto

Ferramenta	Abordagem	Limitação
Spotify API	Estudos e testes de código exemplo	Não é capaz de abrir o Spotify em dispositivos
Google Assistant gRPC API	Estudos e testes de código exemplo	Não é capaz de interagir com outros aplicativos ou dispositivos
Google Assistant Relay	Estudo do código e documentação da ferramenta	Não possui a capacidade de fazer a invocação de multimídia
Fulfillment	De acordo com a documentação permite o controle de dispositivos, não testado por falta de tempo	

Tabela 1. Abordagens testadas para controle dos dispositivos de áudio.

Após os estudos feitos destas ferramentas, foi possível se determinar que a API do *Spotify* não permite que a aplicação seja inicializada nos dispositivos. Quanto às APIs *Google Assistant gRPC API* e a *Google Assistant Relay*, estas apresentaram limitações para o uso da *Google Assistant* em um computador, em relação ao uso de aplicações externas através das APIs. Para o *Fulfillment*, os estudos determinaram a capacidade e versatilidade da ferramenta; porém, por falta de tempo, não foi possível a execução de testes para se averiguar se a ferramenta seria adequada para a implementação do sistema proposto na Seção 3.

5. Conclusão

Este trabalho propôs uma arquitetura para o desenvolvimento de sistema de automação doméstica que permitisse a troca de dispositivos multimídia de acordo com a movimentação do usuário. O sistema identifica a posição do usuário, coordenando ações nos dispositivos para que apenas o dispositivo mais perto do usuário fosse utilizado e, à medida que o usuário se deslocasse pelo ambiente, os dispositivos iam sendo ligados e desligados automaticamente.

Após definida a arquitetura, iniciou-se um estudo de viabilidade das ferramentas e APIs que pudessem ser usadas na implementação da proposta. Primeiramente foi testada a API do aplicativo *Spotify*, onde averiguou-se ser possível controlar o áudio, mas não era possível iniciar o aplicativo nos dispositivos. Testou-se também o *Google Assistant*

gRPC API e a *Google Assistant Relay*, que também apresentaram limitações para controlar dispositivos e aplicações externas. Por fim, estudou-se o *Fulfillment*, uma terceira abordagem da *Google* que, de acordo com a documentação, permite controlar dispositivos em um ambiente inteligente. Porém, por falta de tempo, não foi possível testar essa abordagem.

Este projeto deixa em aberto um possível ponto de partida para trabalhos futuros: o da construção e teste de um *Fulfillment*, com as funcionalidades necessárias e a procura por outras ferramentas de controle de dispositivos que permitam a criação de sistema baseado na arquitetura apresentada.

Referências

Al-Kuwari, M., Ramadan, A., Ismael, Y., Al-Sughair, L., Gastli, A., and Benammar, M. (2018). Smart-home automation using iot-based sensing and monitoring platform. In *2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018)*, pages 1–6. IEEE.

Alaa, M., Zaidan, A. A., Zaidan, B. B., Talal, M., and Kiah, M. L. M. (2017). A review of smart home applications based on internet of things. *Journal of Network and Computer Applications*, 97:48–65.

Alletto, S., Cucchiara, R., Del Fiore, G., Mainetti, L., Mighali, V., Patrono, L., and Serra, G. (2015). An indoor location-aware system for an iot-based smart museum. *IEEE Internet of Things Journal*, 3(2):244–253.

Chen, Y.-H., Tsai, M.-J., Fu, L.-C., Chen, C.-H., Wu, C.-L., and Zeng, Y.-C. (2015). Monitoring elder's living activity using ambient and body sensor network in smart home. In *2015 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2962–2967. IEEE.

Kodali, R. K., Jain, V., Bose, S., and Boppana, L. (2016). Iot based smart security and home automation system. In *2016 international conference on computing, communication and automation (ICCCA)*, pages 1286–1289. IEEE.

Mackensen, E., Lai, M., and Wendt, T. M. (2012). Bluetooth low energy (ble) based wireless sensors. In *SENSORS, 2012 IEEE*, pages 1–4. IEEE.

Mainetti, L., Mighali, V., and Patrono, L. (2015). An iot-based user-centric ecosystem for heterogeneous smart home environments. In *2015 IEEE International Conference on Communications (ICC)*, pages 704–709. IEEE.

Moreno, M. V., Ramos, J. L. H., and Skarmeta, A. F. (2014). User role in iot-based systems. In *2014 IEEE World Forum on Internet of Things (WF-IoT)*, pages 141–146. IEEE.

Yachir, A., Amirat, Y., Chibani, A., and Badache, N. (2015). Event-aware framework for dynamic services discovery and selection in the context of ambient intelligence and internet of things. *IEEE Transactions on automation science and engineering*, 13(1):85–102.

Yin, Z., Che, Y., and He, W. (2015). A hierarchical group control method of electrical loads in smart home. In *2015 6th International Conference on Power Electronics Systems and Applications (PESA)*, pages 1–6. IEEE.