Desenvolvimento de Aplicação para um Estacionamento Inteligente via Computação em Nuvem baseada em Agentes e Sistema Ciber-Físico *

Pedro W. Botelho[1], Gleifer Vaz Alves[1], Paulo Leitão[2], André Pinz Borges[1]

[1]Departamento Acadêmico de Informática –
Universidade Tecnológica Federal do Paraná (UTFPR)
Rua Doutor Washington Subtil Chueire, 330 - 84017-220 – Ponta Grossa – PR – Brasil

[2]Research Centre in Digitalization and Intelligent Robotics (CeDRI)
Instituto Politécnico de Bragança
Campus de Santa Apolónia, 5300-253 – Bragança – Portugal

pbotelho@alunos.utfpr.edu.br, pleitao@ipb.pt

{gleifer, apborges}@utfpr.edu.br

Abstract. *Systems for managing and allocating parking spaces aim to reduce urban traffic and improve the occupancy of parking spaces. Some applications used, in isolated forms, Multi-Agent Systems (MAS), Internet of Things (IoT) and Cloud Computing (CC). Integrating these technologies is critical to build smart parking applications. This article presents an implementation of an architecture that integrates MAS technologies for the allocation of vacancies, IoT for detecting cars and a cloud based database for storing information. The integration took place through a cloud API communicating a mobile application to the MAS, the physical device and the database which was proved to be efficient to communicate the devices.*

Resumo. *Sistemas para gerenciamento e alocação de vagas de estacionamentos são desenvolvidos para reduzir tráfego urbano e melhorar a ocupação de estacionamentos. Aplicações desenvolvidas utilizam, de maneira isolada, Sistemas Multi-Agentes (SMA), Internet das Coisas (IoT), e Computação em Nuvem (CN). Integrar tais tecnologias é fundamental para a criação de aplicativos inteligentes para estacionamentos. Aqui é apresentado a implementação de uma arquitetura que integra as tecnologias de SMA para alocação de vagas, IoT para detecção de carros e um banco de dados na nuvem para o armazenamento de informações. A integração deu-se por meio de uma API comunicando um aplicativo móvel ao SMA, ao dispositivo físico e ao banco de dados, a qual se mostrou eficiente para comunicar os dispositivos.*

*O trabalho Desenvolvimento de Aplicação para um Estacionamento Inteligente via Computação em Nuvem baseada em Agentes e Sistema Ciber-Físico de Pedro Warmling Botelho, Gleifer Vaz Alves, Paulo Leitão, André Pinz Borges está licenciado com uma Licença Creative Commons - Atribuição-NãoComercial-CompartilhaIgual 4.0 Internacional. http://creativecommons.org/licenses/by-nc-sa/4.0/

1. Introdução

Problemas de trânsito são encontrados em cidades dos mais diversos portes, desde pequenas até grandes cidades. Estudos mostram que cerca de um terço dos veículos circulando em uma cidade são motoristas em busca de estacionamento [Jazdi 2014]. O método mais comum de busca por vagas é manual, sem auxílio de sistemas computacionais. A solução menos eficiente para esse problema é aumentar o número de vagas, assim os motoristas terão maiores possibilidades de encontrar um estacionamento. Contudo, há métodos inteligentes e, consequentemente, mais eficientes para atenuar ou até resolver o problema, como os estacionamentos inteligentes [Geng and Cassandras 2013].

Estacionamento Inteligente (EI) é um sistema de estacionamento que auxilia os motoristas a encontrar vagas utilizando sensores que detectam se há ou não um veículo e então o direcionam para a vaga [Hassoune et al. 2016]. O objetivo desses sistemas é garantir que o motorista encontre uma vaga para estacionar no local desejado rapidamente. Assim, o motorista pode reservar uma vaga antes mesmo de entrar no veículo [Castro et al. 2017].

No projeto *Smart Parking* (SP) desenvolvido em parceria pelo Instituto Politécnico de Bragança (IPB) com a Universidade Tecnológica Federal do Paraná campus Ponta Grossa (UTFPR-PG), diferentes abordagens foram implementadas neste domínio, como a negociação de vagas em um estacionamento inteligente [Ducheiko et al. 2018] [Alves et al. 2019]. O projeto, além da negociação, aborda também a implementação de um estacionamento inteligente como um todo, contendo sistemas físicos, também já implementados [Botelho et al. 2019] [Sakurada et al. 2019]. Porém, o projeto ainda não conta com uma aplicação que o usuário possa utilizar para fazer a reserva de vagas e também não conta com um meio de integrar todos estes componentes, sendo estes itens propostos aqui.

Para solucionar o problema dos estacionamentos inteligentes, foi adotado, pelo projeto, uma abordagem baseada em agentes devido a capacidade de inteligência distribuída dos agentes e pela escalabilidade oferecida [Wooldridge 2009]. A arquitetura estabelece dois tipos de agentes: o agente motorista, o qual representa a pessoa que dirige o veículo e o agente vaga que representa uma vaga de estacionamento. Neste contexto, os agentes negociam uma vaga até chegar em um consenso e só então o motorista pode se deslocar ao estacionamento.

Desse modo, com o número da vaga já de posse do motorista, é necessário um sistema físico para realizar o controle da vaga e dos setores do estacionamento, como detectar quando um carro é estacionado. Porém, atualmente no projeto, ainda não é possível fazer com que uma requisição do usuário seja transformada em uma negociação pelo Sistema Multi-Agente (SMA) e, consequentemente, no ato de detectar o carro estacionado na vaga.

Para alcançar tal objetivo, é necessário implementar uma interface que permita ao motorista requisitar uma vaga e integrar esta interface ao SMA e a parte ciber-física. Para isso, neste trabalho é implementado um aplicativo móvel e uma solução em nuvem para permitir que os componentes de hardware troquem mensagens e realizem o controle das vagas do estacionamento com apoio de tecnologia em nuvem. Cabe destacar que o foco do artigo não está no processo de negociação, uma vez que foram utilizados mecanismos

de negociação simples, com valores iniciais aleatório e sem a preocupação com tempo de negociação.

O restante deste artigo está organizado da seguinte forma. Na seção 2 são descritos alguns trabalhos relacionados ao tema de Sistemas Multi-Agentes, plataformas de hardware e computação em nuvem. Na seção 3, tem-se a descrição da arquitetura implementada no artigo e como cada componente se comporta dentro dela. A seção 4 apresenta os resultados obtidos e a seção 5 as considerações finais a respeito do desenvolvimento do trabalho.

2. Trabalhos Relacionados

Nesta seção descreve-se brevemente alguns trabalhos desenvolvidos pelo grupo de pesquisa e alguns que relacionam estacionamentos inteligentes com componentes físicos e com computação em nuvem.

Em [Ducheiko et al. 2018], é proposto um modelo de raciocínio e protocolo de negociação para alocação de vagas em um estacionamento utilizando um mecanismo de negociação descentralizado. Neste mecanismo, existe apenas um tipo de agente no sistema, o agente motorista, o qual pode assumir o papel de vendedor (quando deixa a vaga) ou comprador (quando requisita a vaga).

O trabalho de [Pham et al. 2015] propõe um sistema de estacionamento que auxilia o usuário a encontrar automaticamente, sem o uso de agentes, uma vaga livre calculando o preço com base na distância do usuário e o número total de vagas gratuitas em cada estacionamento. Este artigo utiliza o conceito da nuvem de forma implícita para realizar a integração entre vários estacionamentos espalhados pela cidade, assim como um aplicativo móvel que permite ao usuário requisitar uma vaga.

Em [Alves et al. 2019] é apresentado um estudo a respeito de protocolos de negociação para problemas de consenso em estacionamentos inteligentes. Algumas estratégias de negociação, como o protocolo *Contract Net*, o leilão inglês e o leilão holandês foram implementados utilizando o *framework* JADE para evidenciar a melhor estratégia para aplicar em um SMA aplicado à estacionamentos inteligentes.

Em [Khanna and R. Anand] um sistema de estacionamento é proposto também a computação em nuvem para integrar os componentes de IoT que detecta a presença de um veículo e um dispositivo móvel para realizar as requisições do usuário. O sistema fornece informações em tempo real do estado da vaga e permite ao motorista escolher uma vaga livre apenas, sem passar por um processo de negociação.

Em [Sakurada et al. 2019] uma arquitetura ciber-física para estacionamentos inteligentes é apresentada, porém, esta arquitetura considera um estacionamento para bicicletas. Cada vaga possui um Raspberry Pi com um agente embarcado, o qual controla um dispositivo físico capaz de liberar o acesso a vaga da bicicleta.

Como foi observado, os trabalhos citados utilizam de maneira isolada diferentes componentes que fazem parte de um estacionamento inteligente como um todo. A partir disso, uma integração entre todos estes componentes é necessária para a construção do aplicativo móvel para estacionamentos inteligentes.

Um trabalho externo ao grupo de pesquisa, desenvolvido por [Bar 2019], apre-

senta uma revisão da literatura sobre tecnologias (e.g. componentes, sensores e softwares) empregadas em Estacionamentos Inteligentes. Os autores identificam os tipos mais utilizados de cada componente e destacam as tendências de uso, além de analisar vários trabalhos cujo foco está no desenvolvimento de softwares para Estacionamentos Inteligente. Este trabalho é utilizado como referência quanto às tecnologias e componentes que podem ser utilizados neste artigo.

3. Metodologia

A computação em nuvem está em uma crescente nos últimos anos e vem auxiliando empresas a se tornarem mais responsivas e ágeis, podendo ser acessada de qualquer lugar e em qualquer dispositivo [Senyo et al. 2018]. Nesse contexto, a arquitetura desenvolvida neste artigo é baseada na possibilidade de conectar componentes de modo independente. Estes componentes vão desde sensores capazes de informar se há a presença de um veículo em uma vaga (e.g. infra-vermelho), dispositivo de comunicação destes sensores com Raspberry Pi (e.g. ESP), Arduíno, aplicativo móvel até serviços em nuvem (e.g. *Amazon Web Services - AWS*).

Uma visão geral da arquitetura e componentes utilizados é apresentada na Figura 1. A arquitetura contém: (i) um aplicativo móvel para a conexão do motorista com o estacionamento; (ii) um módulo de computação em nuvem (AWS) usado para replicar mensagens trocadas entre os componentes e armazenar todas as informações do estacionamento; (iii) um SMA responsável pela negociação da vaga requisitada pelo motorista e (iv) a plataforma de hardware é responsável por detectar a presença de um veículo na vaga.

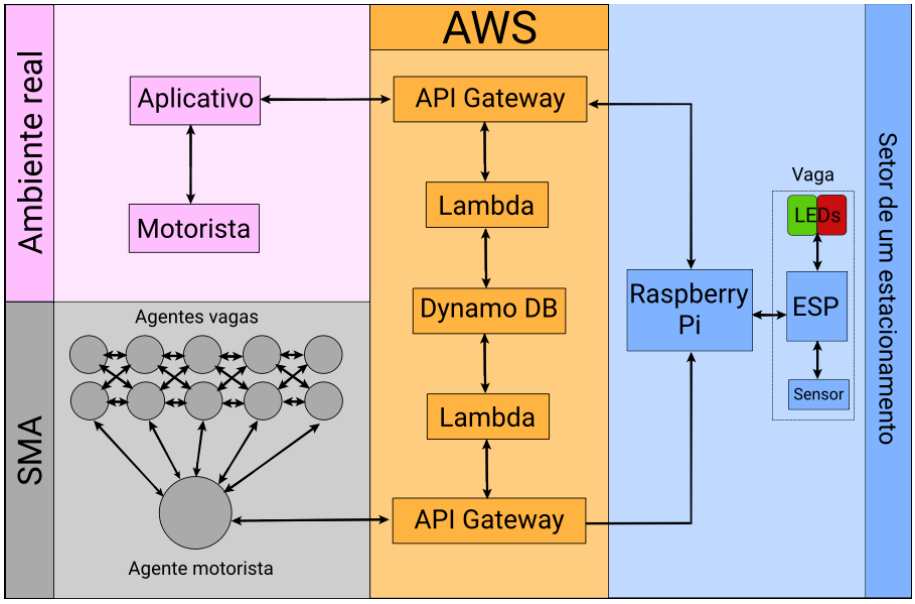

Figura 1. Visão geral da arquitetura

De maneira geral, o sistema inicia quando um motorista requer uma vaga de estacionamento pelo aplicativo móvel (Aplicativo). Este se comunica com a nuvem (AWS) a qual repassa esta informação ao SMA. O SMA realiza uma negociação com base nos critérios informados pelo motorista e devolve para a nuvem o setor e o número da vaga em

que o motorista deve estacionar. Neste momento, o motorista é registrado como locatário da vaga no banco de dados e o setor e o número da vaga são repassados ao aplicativo, o qual informa ao motorista que ele deve se dirigir a vaga. O motorista, por sua vez, estaciona na vaga e então, a plataforma de hardware vinculada a vaga reconhece o carro do motorista e armazena na nuvem os dados necessários.

3.1. Aplicativo Smart Parking

Para o desenvolvido do aplicativo, o framework Flutter [Flutter 2017] foi utilizado por possuir uma metodologia de desenvolvimento híbrido capaz de compilar o aplicativo para Android e iOS mantendo apenas uma base de código. Assim, o aplicativo une o usuário e o restante do sistema com uma interface de fácil uso, uma vez que consiste em poucas funcionalidades, como cadastrar um novo veículo, estacionar e consultar seu histórico de estacionamento.

Para requisitar uma vaga, o motorista deve, primeiro, cadastrar a placa do carro em e então requisitar uma vaga, onde ele irá visualizar as vagas disponíveis e selecionar uma, informando o preço que deseja pagar na vaga. Como dito anteriormente, neste trabalho os valores a pagar pelas vagas, tempo de estacionamento, quantidade de vagas, tempo de permanência e valores relativos à negociação são gerados apenas para simulação. Ao término da requisição (processo descrito na seção anterior) o aplicativo notifica o motorista se a vaga requerida por ele foi, ou não, alocada para ele. Nota-se que uma outra vaga, que seja parecida com a escolhida por ele pode ser alocada, essa informação dar-se-à pela implementação do SMA.

3.2. Sistema Multi-Agente

Uma parte importante do sistema é a negociação entre o motorista e a vaga. No entanto, a implementação deste componente foi realizado por outra parte do projeto e integrado à esta arquitetura (ver [Alves et al. 2019]). Portanto, considera-se que, quando uma vaga for requisitada, um agente assumirá o papel de motorista e irá conter os requisitos da vaga informadas pelo usuário no momento da escolha da vaga, outro agente assumirá o papel de vaga e ambos irão negociar uma vaga até que cheguem em um consenso. Este conjunto de requisitos leva em consideração a vaga escolhida pelo motorista.

A implementação da negociação pode ocorrer seguindo três abordagens, o Protoloco Contract Net (CNP), o leilão holandês e o leilão inglês. O CNP é basicamente uma licitação, onde o agente motorista envia uma mensagem para todos os agentes contendo seus requisitos e todos os agentes vagas que possuem algum dos requisitos enviam ao agente motorista suas características, assim, o agente motorista decide qual das vagas possui mais características baseado nos seus requisitos.

O leilão inglês tenta obter um consenso entre o agente motorista e o agente vaga mudando o preço a cada rodada, começando com um preço baixo até que nenhum dos agentes vagas aceite mais mudar o preço. Já o leilão holandês começa com um preço alto e vai baixando até que um agente vaga aceite.

3.3. Plataforma de Hardware

A Plataforma de Hardware é um componente fundamental para o correto funcionamento da arquitetura, pois tem a funcionalidade de detectar se um carro foi estacionado e alertar

os outros componentes (da arquitetura) para alteração do status da vaga. A implantação desta plataforma foi realizada em [Botelho et al. 2019], a qual possui um agente desenvolvido em JADE embarcado em um Raspberry Pi[1] gerenciando um setor de um estacionamento com n vagas e cada vaga possui um ESP-12e conectado a sensores de presença e a dois LEDs de cores vermelho (*indicando vaga ocupada*) e verde (*representando vaga livre*).

O sensor de presença localizado em cada vaga permite que o ESP-12e receba um sinal quando um carro é estacionado na vaga, assim, é possível enviar uma mensagem ao Raspberry Pi, informando-o que a vaga está ocupada. Esta comunicação entre os dispositivos é realizada através do protocolo MQTT [MQTT 1999] que utilizando o princípio *Publish-Subscribe*, permite a troca de mensagens por meio de tópicos entre dispositivos conectados à Internet.

Quando é enviada a mensagem: "existe um carro estacionado" ao Raspberry Pi, o agente realiza uma comunicação com o serviço em nuvem, informando que o status da vaga estacionada deve mudar para ocupado. Após obter a resposta positiva do serviço em nuvem, o agente envia ao ESP-12e uma mensagem indicando que o LED vermelho deve ser ligado. Caso receba uma resposta negativa, ou seja, o status da vaga estacionada não é um status de negociação, o LED vermelho começa a piscar.

O processo relacionado a quando o carro deixa a vaga é semelhante. O agente recebe a mensagem que o carro saiu da vaga e repassa a mesma para o serviço em nuvem, o qual irá atualizar o status da vaga para livre, habilitando novamente a vaga para negociação. Após receber uma mensagem positiva da nuvem, o agente informa o ESP-12e que o LED verde pode ser ligado e o vermelho desligado.

3.4. Computação em Nuvem

Para realizar a ligação entre todos os componentes citados nas sub-seções anteriores, tem-se um conjunto de micro-serviços hospedados na *Amazon Web Services* (AWS)[2]. O primeiro deles é o DynamoDB[3], um banco de dados não relacional que armazena as informações do motorista e das vagas. Todos os setores e as vagas do estacionamento estão mapeados no banco de dados, contendo informações sobre o motorista que ocupa a vaga, como a placa do carro estacionado, e sobre a própria vaga, se ela está livre, ocupada ou em negociação. No banco de dados, também são armazenados os dados do motorista no momento do cadastro no sistema e um histórico de todas as vezes que o motorista estacionou.

O segundo micro-serviço utilizado é a *API Gateway* [4] com comunicação via HTTP e via *WebSocket*, criando uma conexão entre o aplicativo, o SMA e o controlador físico, enviando os dados compartilhados por eles dentro do sistema. Quando uma mensagem é enviada por um dos componentes para a API, um terceiro micro-serviço é acionado, chamado de *função Lambda*. Esta função é uma rotina, escrita utilizando a linguagem GO [Golang 2009], que executa diferentes tipos de ações dependendo de quem a chama. Basicamente, existe uma Lambda para cada processo executado no sistema, como criar

[1] www.raspberrypi.org
[2] https://aws.amazon.com/
[3] https://aws.amazon.com/pt/dynamodb/
[4] https://aws.amazon.com/pt/api-gateway/

usuário no banco, atualizar o status da vaga, enviar a requisição do usuário para o SMA, entre outros.

A seguir é descrito o processo de como cada função é executada de modo a informar o motorista o número do setor e da vaga a ser estacionada.

1. O motorista clica no botão de estacionar;
2. Uma requisição HTTP é feita para o *API Gateway*, o qual executa uma função *Lambda* que busca no banco de dados *DynamoDB* as vagas livres e ocupadas do sistema e retorna para o aplicativo;
3. O usuário visualiza as vagas e seleciona a que melhor se encaixa nos seus requisitos;
4. Uma requisição via *WebSocket* é feita para o *API Gateway*, o qual executa uma função *Lambda* que replica a mensagem para o SMA. A mesma função atualiza o status da vaga para em negociação. O controlador físico também recebe esta mensagem e o LED de cor verde fica piscando;
5. O SMA realiza a negociação da vaga e realiza uma requisição, via *WebSocket*, para o *API Gateway*, que executa uma função que envia para o aplicativo o número da vaga negociada;
6. O motorista recebe uma notificação de que a vaga foi negociada, neste momento ele pode aceitar ou não a vaga. Caso aceite, uma requisição HTTP é feita para o *API Gateway*, vinculando os dados do motorista àquela vaga. Caso não aceite, uma requisição via *Web Socket* é feita, informando o controlador físico que o LED de cor verde pode parar de piscar;
7. O motorista agora pode se dirigir a vaga.

A parte física se comunica com a nuvem de modo semelhante: após o processo descrito na sub-seção anterior, o Raspberry Pi responsável pelo setor é quem realiza as chamadas HTTP para o *API Gateway*, o qual executa uma função Lambda para atualizar o status da vaga no banco de dados.

4. Resultados

Para analisar os resultados, uma vaga foi requisitada no aplicativo, então, esperou-se até obter a resposta da negociação da vaga, uma vez que o sucesso deste processo exemplifica todo os passos descritos anteriormente.

A Figura 2 ilustra os setores e as vagas do estacionamento, cada círculo representa uma vaga e é necessário clicar em uma bola verde para requisitar uma vaga, nota-se que bolas vermelhas são ocupadas e azuis são em negociação. A Figura 3 ilustra a mensagem que uma vaga foi requisitada. Neste momento a comunicação do aplicativo com a nuvem é realizada, repassando a informação ao SMA que inicia a negociação.

O processo de negociação descrito neste artigo consiste da verificação da disponibilidade da vaga e se há saldo para efetuar o pagamento. Um dos próximos passos da pesquisa é incorporar, neste trabalho, um método de negociação desenvolvido neste projeto, por exemplo [Alves et al. 2019, Ducheiko et al. 2018].

A Figura 4 ilustra a mensagem que a vaga requisitada foi negociada e atribuída ao motorista com êxito e a Figura 5 ilustra o histórico de estacionamento daquele motorista no sistema.

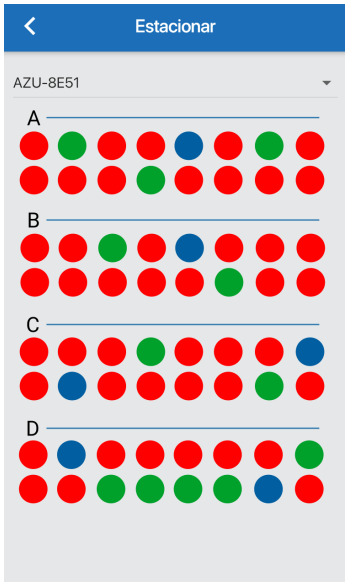

Figura 2. Escolha de vaga

Figura 3. Vaga requisitada

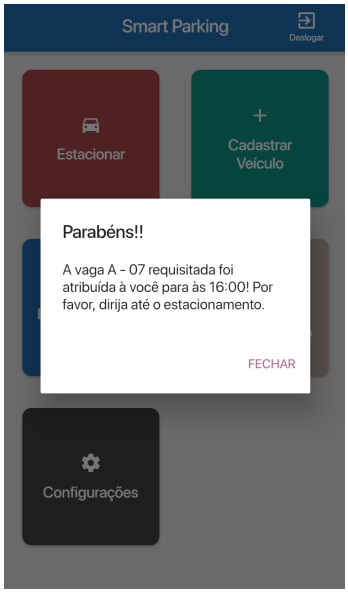

Figura 4. Resultado da requisição

Figura 5. Histórico de estacionamento

5. Conclusões

Este artigo apresenta a implementação de um aplicativo móvel e uma API na nuvem como uma solução para integrar os componentes presentes em um estacionamento inteligente. A arquitetura possui como uma de suas características a versatilidade, sendo possível utilizar qualquer dispositivo móvel para realizar uma requisição. Outra característica diz respeito a facilidade da comunicação entre os dispositivos, sendo necessário apenas de acesso a internet para que os componentes possam se comunicar.

Tendo como base os experimentos apresentados na seção anterior, é possível evidenciar que, para o correto funcionamento desta arquitetura, não é necessário implementar exatamente os componentes aqui citados, como o mesmo SMA, ou o mesmo aplicativo.

A computação em nuvem consegue tornar esta arquitetura flexível, o que significa que um outro SMA, utilizando outro protocolo de negociação pode ser utilizado, desde que receba como entrada uma vaga para negociar e retorne como saída uma vaga negociada.

Como continuidade imediata deste trabalho, deseja-se aplicar testes de desempenho nesta aplicação a fim de evidenciar possíveis gargalos de requisições simultâneas e testar a aplicabilidade desta arquitetura como um todo em um cenário com componentes de um ambiente físico.

Referências

(2019). Smart parking: A literature review from the technological perspective. *Applied Sciences (Switzerland)*, 9(21).

Alves, B. R., Alves, G. V., Borges, A. P., and Leitão, P. (2019). Experimentation of Negotiation Protocols for Consensus Problems in Smart Parking Systems. In Mařík, V., Kadera, P., Rzevski, G., Zoitl, A., Anderst-Kotsis, G., Tjoa, A. M., and Khalil, I., editors, *Industrial Applications of Holonic and Multi-Agent Systems*, Lecture Notes in Computer Science, pages 189–202, Cham. Springer International Publishing.

Botelho, P. W., Alves, G. V., and Borges, A. P. (2019). Proposta de implantação de um sistema ciber-físico para um smart parking baseado em agentes inteligentes. In *Workshop-School on Agents, Environments, and Applications*, pages 259–264.

Castro, L. F. S. D., Alves, G. V., and Borges, A. P. (2017). Using trust degree for agents in order to assign spots in a Smart Parking. *ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal*, 6(2):45–55.

Ducheiko, F. F., Alves, G. V., and Borges, A. P. (2018). Implantação de um modelo de raciocínio e protocolo de negociação para um estacionamento inteligente com mecanismo de negociação descentralizado. In *Revista Junior – ICCEE*, pages 25–32.

Flutter (2017). Flutter documentation.

Geng, Y. and Cassandras, C. G. (2013). New "smart parking" system based on resource allocation and reservations. *Proceedings of the IEEE Transactions on Intelligent Transportation Systems*, 14(3):1129–1139.

Golang (2009). Go documentation.

Hassoune, K., Dachry, W., Moutaouakkil, F., and Medromi, H. (2016). Smart parking systems: A survey. In *2016 11th International Conference on Intelligent Systems: Theories and Applications (SITA)*, pages 1–6. IEEE.

Jazdi, N. (2014). Cyber physical systems in the context of industry 4.0. *Proceedings of the IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR'14)*, pages 1–4.

Khanna, A. and R. Anand, journal=International Conference on Internet of Things and Applications (IOTA), t. y. p. m.

MQTT (1999). Mqtt documentation.

Pham, T. N., Tsai, M. F., Nguyen, D. B., Dow, C. R., and Deng, D. J. (2015). A cloud-based smart-parking system based on internet-of-things technologies. *IEEE Access*, 3(1):1581–1591.

Sakurada, L., Barbosa, J., Leitão, P., Botelho, P. W., Alves, G. V., and Borges, A. P. (2019). Development of agent-based cps for smartparking systems. In *IECON - Annual Conference of the IEEE Industrial Electronics Society*.

Senyo, P. K., Boateng, B., and Addae, E. (2018). Cloud computing research: A review of research themes, frameworks, methods and future research directions. *International Journal of Information Management*, pages 128–137.

Wooldridge, M. (2009). *An introduction to multiagent systems*. John Wiley & Sons.