UNIVERSIDADE FEDERAL DE SANTA MARIA
COLÉGIO POLITÉCNICO
CURSO DE GRADUAÇÃO EM TECNOLOGIA EM SISTEMAS PARA INTERNET

Luiz Felipe Romero de Azevedo

UM ESTUDO SOBRE ALTERNATIVAS DE ROAMING EM MOBILIDADE ELÉTRICA

Santa Maria, RS
2024

Luiz Felipe Romero de Azevedo

UM ESTUDO SOBRE ALTERNATIVAS DE ROAMING EM MOBILIDADE ELÉTRICA

Monografia apresentada ao Curso de Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Prof. Dr. Rafael Gressler Milbradt

Santa Maria, RS
2024

[1] *“As aventuras nunca têm fim? Suponho que não. Alguém sempre tem que continuar a história. “*

(J.R.R. Tolkien, 1954)

Luiz Felipe Romero de Azevedo

UM ESTUDO SOBRE ALTERNATIVAS DE ROAMING EM MOBILIDADE ELÉTRICA

Monografia apresentada ao Curso de Sistemas para Internet da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do título de **Tecnólogo em Sistemas para Internet.**

Aprovado em 9 de agosto de 2024:

_____
Rafael Gressler Milbradt, Dr. (UFSM)
(Presidente/Orientador)

_____
Vanessa Gindri Vieira, Msc. (UFSM)

_____
Juçara Salete Gubiani, Dra. (UFSM)

Santa Maria, RS
2024

RESUMO

Este trabalho analisa a infraestrutura de carregamento no contexto da mobilidade elétrica, concentrando-se especificamente na dinâmica da comunicação entre operadores de ponto de carregamento (CPOs) e provedores de serviço de mobilidade elétrica (eMSPs), evidenciando desafios no estabelecimento de uma comunicação consistente e escalável nesse ecossistema em expansão. É destacado a importância do roaming em mobilidade elétrica, ressaltando como ele desempenha um papel determinante no crescimento contínuo desse setor. O estudo se aprofunda na compreensão do protocolo *Open Charge Point* Interface (OCPI) explorando sua funcionalidade como uma solução para a falta de interoperabilidade entre CPOs e eMSPs. Também é visto o papel de um roaming hub na superação da dificuldade de escalabilidade de comunicações e o porquê de ele ser uma solução.

Palavras-chave: Provedores de serviço de mobilidade elétrica. Operadores de ponto de carregamento. Roaming. Interoperabilidade. Hub. Mobilidade elétrica. Protocolo.

ABSTRACT

This paper analyzes the charging infrastructure in the context of electric mobility, with a particular focus on the dynamics of communication between Charge Point Operators (CPOs) and Electric Mobility Service Providers (eMSPs), highlighting challenges in establishing consistent and scalable communication in this expanding ecosystem.The importance of roaming in electric mobility is highlighted, emphasizing how it plays a critical role in the continued growth of the sector. The study delves into the understanding of the Open Charge Point Interface (OCPI) protocol, exploring its functionality as a solution to the lack of interoperability between CPOs and eMSP. It also looks at the role of a roaming hub in overcoming the challenges of communication scalability and why it serves as a solution.

Keywords: Electric Mobility Service Provider. Charge Point Operator. Interoperability. Roaming. Hub. Electric Mobility. Protocol.

LISTA DE FIGURAS

SUMÁRIO

1. INTRODUÇÃO

O transporte movido à combustão é um dos principais contribuintes para a poluição do ar. Tendo em vista a diminuição da emissão de gases poluentes e, consequentemente, uma melhora na qualidade de vida dos seres humanos, a não utilização de combustíveis fósseis neste setor é um meio para alcançar esse objetivo. O aumento de vendas/emplacamentos de veículos eletrificados no Brasil de 2012 a dezembro de 2022 foi de 42.900% (ABVE, 2022), evidenciando um grande crescimento pela busca por veículos elétricos como meio de transporte. A projeção para 2030 é de um aumento médio anual de 30% da frota de veículos elétricos desde 2022 (Global Electric Vehicle Outlook, 2023).

Com o crescimento global da frota de veículos elétricos, faz-se necessário o aumento quantitativo de estações de carregamento (EVCS – *Electric Vehicle Charging Station*), a fim de proporcionar segurança e confiabilidade aos proprietários. Desse modo, tendo um papel de suma importância no estímulo da adoção da mobilidade elétrica. Entretanto, junto ao crescimento no número de EVCSs é necessário um meio de comunicação entre os proprietários desses veículos com o carregador de um EVCS, com o objetivo de viabilizar a operabilidade do carregamento.

O gerenciamento de um EVCS é realizado por operadores, cuja responsabilidade principal é garantir uma comunicação eficiente com os carregadores. As informações sobre os EVCSs necessitam ser acessíveis aos proprietários de veículos elétricos, o que requer uma interação entre softwares para o fornecimento e a obtenção desses dados tendo em vista que o aplicativo utilizado para a visualização das informações precisa dos dados do carregador para exibir ao usuário final.

Para estabelecer essa comunicação, é necessário que esses sistemas compreendam a estrutura das interfaces de programação oferecidas pelas estações de carregamento. No entanto, essa abordagem pode levar a complexidades adicionais, aumentando os custos e dificultando a expansão para suporte a EVCSs por parte desses aplicativos. A falta de informações sobre EVCSs tende a causar *range anxiety,* que segundo Driivz Team (2023), é o medo que proprietários de veículos elétricos possuem de que a bateria do seu veículo não será suficiente para determinado trajeto e que carregadores podem não estar disponíveis no momento que precisam. Dessa forma, a falta de suporte a informações sobre EVCSs devido à

complexidade de implementação pode causar o fenômeno descrito anteriormente em seus usuários e consequentemente diminuir à atratividade a adoção de veículos elétricos.

Garantir o uso de qualquer carregador a partir de qualquer aplicativo não é uma tarefa fácil, pois tanto o número de aplicativos como o número de estações de carregamento são grandes e tendem a crescer cada vez mais. A falta de padronização na comunicação entre serviços de carregamento tem sido um desafio significativo que afeta negativamente a experiência dos usuários e a expansão da mobilidade elétrica como um todo.

Os proprietários de veículos elétricos enfrentam obstáculos consideráveis, uma vez que a comunicação entre diferentes aplicativos e estações de carregamento muitas vezes não existe. Isso pode levar a uma série de problemas, como a necessidade de manter vários aplicativos diferentes para acessar diferentes estações de carregamento, dificuldades na localização de carregadores disponíveis e no controle do estado atual de sua utilização.

Esses desafios são uma barreira para adoção ampla de veículos elétricos. Dessa forma, é fundamental abordar esses desafios e encontrar soluções que permitam a comunicação eficiente e a interação harmoniosa entre todos os atores envolvidos na infraestrutura de carregamento.

A fim de proporcionar dinamicidade ao proprietário de veículo elétrico, possibilitando alta escalabilidade de EVCSs por aplicativos e padronizando as informações e operações do software que gerencia um EVCS independente do aplicativo utilizado pelo usuário, o *Open Charge Point Interface* (OCPI) é um protocolo que pode ser adotado para atingir esse objetivo através da padronização das interfaces de programação oferecidas pelas estações de carregamento e aplicativos.

O protocolo OCPI surge como uma solução potencial para padronizar a comunicação entre serviços de carregamento. Através do estudo do funcionamento deste protocolo e de como ele pode ser implementado para superar as barreiras geradas pela falta de interoperabilidade, este trabalho busca contribuir para a promoção de uma infraestrutura de carregamento mais eficiente, acessível e confortável aos usuários.

O OCPI define padrões para a troca de informações entre serviços de carregamento, sua adoção pode simplificar essa comunicação visando a escalabilidade, sendo assim, aumentando a gama de estações de carregamento

disponíveis por aplicativo, permitindo que os usuários encontrem, reservem e utilizem carregadores de forma mais conveniente e sem as limitações impostas pela falta de interoperabilidade.

Portanto, a investigação do OCPI neste contexto é fundamental para o avanço da mobilidade elétrica e a construção de um futuro mais sustentável e eficiente em termos de transporte.

1.1 CONTEXTUALIZAÇÃO DO PROBLEMA

1.1.1 Problema

A alta demanda por melhorias na infraestrutura de carregamento para veículos elétricos no Brasil é evidente, com uma infraestrutura sólida e madura a tendência é que haja um aumento de veículos elétricos nas ruas, pois os cidadãos poderão ter mais tranquilidade uma vez que o *range anxiety* for diminuído. Reduzir o *range anxiety* é crucial para incentivar a adoção de veículos elétricos e promover uma mobilidade mais limpa e sustentável.

O problema central reside na dificuldade de estabelecer uma comunicação eficaz entre diferentes operadores de estações de carregamento e fornecedores de serviços de carregamento. A ausência de padrões claros para a troca de informações cria barreiras significativas tanto para os serviços de carregamento quanto para os usuários finais desses serviços. Esta falta de interoperabilidade gera um problema que é a dificuldade de usuários em usar diferentes redes de carregamento, resultando em uma experiência inconsistente, desconfortável e muitas vezes frustrante.

Portanto, este estudo busca aprofundar a compreensão dos desafios associados ao roaming no âmbito da mobilidade elétrica e abordar com foco o OCPI, destacando-o como uma solução potencial para mitigar esses desafios, visando promover uma mobilidade elétrica mais eficiente e viável.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Analisar a comunicação entre CPOs e eMSPs, investigando seus desafios no contexto da infraestrutura de carregamento para veículos elétricos.

1.2.2 Objetivos Específicos

- Avaliar a importância do roaming na mobilidade elétrica, destacando como essa abordagem influencia o crescimento e a eficiência do setor.
- Aprofundar o entendimento do protocolo OCPI, examinando suas funcionalidades e aplicações como solução para a falta de interoperabilidade entre CPOs e eMSPs
- Analisar o papel do roaming hub na superação dos desafios de escalabilidade de comunicações entre CPOs e eMSPs, identificando as razões pelas quais essa abordagem é considerada uma solução eficaz.
- Implementar o processo de registro do módulo de credenciais do OCPI.

1.3 JUSTIFICATIVA

A busca por alternativas de mobilidade mais sustentável tem se tornado uma prioridade global à medida que as preocupações ambientais e a necessidade de reduzir a poluição do ar se intensificam. A mobilidade elétrica tem se destacado como uma solução promissora, uma vez que os veículos elétricos produzem emissões zero no ponto de uso. Essa transição para veículos elétricos é uma maneira eficaz de reduzir a poluição do ar e diminuir a dependência de combustíveis fósseis.

O *range anxiety* tem sido uma barreira significativa para a escolha por veículos elétricos. A falta de confiança na disponibilidade de carregadores afeta a decisão de compra e age como uma força oposta a consolidação da mobilidade elétrica. Este trabalho contribuirá para a pesquisa acadêmica no campo da mobilidade elétrica, destacando a importância da interoperabilidade entre serviços de carregamento e apresentando o protocolo OCPI como solução para a padronização da comunicação entre serviços de carregamento, desse modo construindo interoperabilidade.

Este capítulo de justificativa estabelece claramente a importância da pesquisa, destacando os problemas a serem abordados e o potencial impacto na promoção da mobilidade elétrica no mundo.

1.4 ORGANIZAÇÃO DO TRABALHO

O capítulo 2 contém o estudo teórico sobre o funcionamento da infraestrutura de comunicação no contexto da mobilidade elétrica. Ele está dividido em 5 subseções, que discutem conceitos, agentes e suas relações nesse contexto, além de apresentar o protocolo OCPI em detalhe, abordando implementações reais em diferentes contextos e exibe o ponto de vista desses diferentes casos.

O capítulo 3 possui o processo de desenvolvimento da implementação do módulo de Credenciais do OCPI e detalha algumas das classes importantes para o desenvolvimento desse módulo, além de conduzir ao funcionamento do código e como devidamente a comunicação nesse módulo ocorre.

Por fim, o capítulo 4 utiliza o que foi abordado no capítulo 3 para a realização de testes, ilustrando o funcionamento da comunicação entre duas plataformas conectadas através do OCPI, mais especificamente, do módulo de Credenciais.

2 REFERENCIAL TEÓRICO

Neste capítulo será explorado em detalhes o protocolo *Open Charge Point Interface (OCPI)*, este protocolo tem o potencial de transformar a maneira que carregamentos de veículos elétricos funcionam em grande parte das estações de carregamento no mundo todo, tornando a experiência de carregamento mais conveniente e confortável para o usuário. No âmbito da constante evolução da mobilidade elétrica, o "roaming" se destaca como uma técnica utilizada pelo OCPI que permite aos proprietários de veículos elétricos carregarem-nos em diferentes carregadores, independentemente do aplicativo utilizado. Este capítulo aprofunda a compreensão sobre esse conceito, destacando suas complexidades e impactos, além de citar agentes cruciais da indústria da mobilidade elétrica. O documento de referência que serviu como base para as definições do OCPI e suas especificações neste capítulo é o OCPI 2.2 publicado pela EV Roaming Foundation, (OCPI, 2023).

Os dois principais agentes explorados nesse capítulo são: *Charge Point Operator* (CPO) e *Electric Mobility Service Provider* (eMSP). O gerenciamento de um EVCS é realizado por um CPO, que, por sua vez, tem como função comunicar-se com o carregador e obter informações em tempo real sobre ele (DRIIVZ TEAM, 2023). Em seguida, as informações sobre os carregadores precisam ser de conhecimento dos proprietários de veículos elétricos. A fim de que isso seja viabilizado, softwares que permitem usuários visualizarem informações e operarem o carregador necessitam se comunicar com os CPOs com o propósito de obterem esses dados. Esses softwares são aplicativos que o usuário poderá utilizar para carregar seu veículo e são também conhecidos como eMSPs.

2.1 MOBILIDADE ELÉTRICA E ROAMING

De acordo com Bekkers e Van der Kam, roaming no contexto de mobilidade elétrica significa que o carregamento de um veículo elétrico através de um eMSP possa ser realizado em qualquer EVCS mesmo que o proprietário do veículo não tenha um contrato direto com o CPO. Porém o eMSP utilizado possuirá um contrato com o CPO diretamente ou através de um *roaming hub* (2020).

Segundo uma pesquisa realizada pelo IEA (2021), a barreira mais significativa para a adoção de um veículo elétrico foi a falta de infraestrutura de carregamento (67%). Como resultado da adoção de roaming na mobilidade elétrica, é esperado uma melhora na infraestrutura de carregamento, já que o usuário não precisará se

preocupar se a estação de carregamento, que ele vai realizar a recarga do seu veículo, é suportada pelo eMSP utilizado. Uma vez que o usuário tem a disponibilidade de uma gama ampla de estações de carregamento disponíveis, a autonomia do veículo elétrico diminuirá sua relevância na preferência das pessoas, pois o carregamento do seu veículo poderá ser realizado em muitas estações de carregamento.

Visto que a utilização de roaming possibilita o carregamento de veículos elétricos em qualquer estação de carregamento operada por um CPO através de um único eMSP, isto facilitará muito a recarga, logo é de suma importância para a adoção da mobilidade elétrica uma vez que proporciona confiabilidade e segurança para o motorista através da operabilidade de carregadores gerenciados por CPOs independente do eMSP utilizado.

2.2 COMUNICAÇÃO ENTRE SERVIÇOS DE CARREGAMENTO

Para começar a abordagem sobre a comunicação entre serviços de carregamento, a seguir será contada uma história ilustrativa. Nesta narrativa, serão estabelecidos personagens, definido agentes, suas responsabilidades e interação no contexto da mobilidade elétrica.

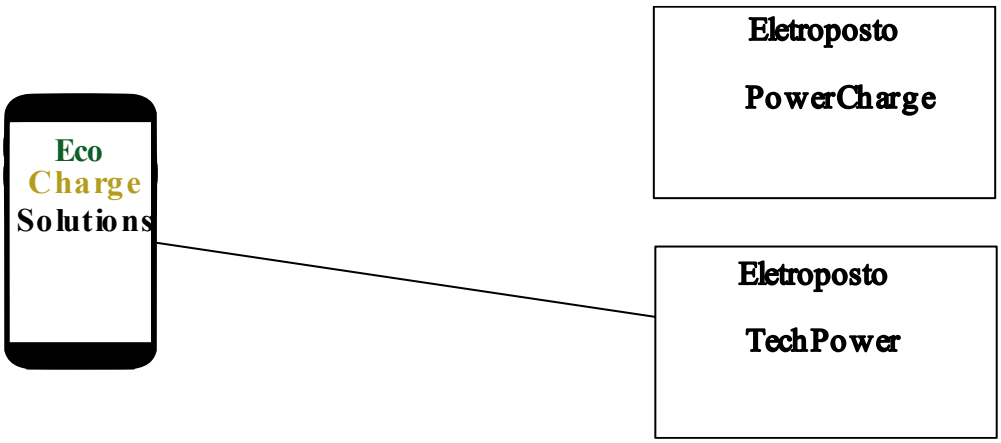
Luiz, um usuário da mobilidade elétrica, que possui um veículo elétrico, tem em seu celular o aplicativo “EcoCharge Solutions”, este é o aplicativo que Luiz utiliza para localizar eletropostos na cidade em que reside, Santa Maria. O eletroposto “TechPower” foi conhecido por Luiz através do “EcoCharge Solutions”, e passou a ser seu lugar de confiança para realizar as recargas de seu veículo.

Durante uma viagem de carro a outra cidade, Luiz se depara com a situação de que necessita realizar a recarga do seu veículo, por sorte ele avista o eletroposto “PowerCharge” e vai até ele para efetuar a recarga. Como de costume, Luiz abre seu aplicativo de confiança, o “EcoCharge Solutions” para começar o carregamento, porém dessa vez ele encontra um desafio inesperado, seu aplicativo de confiança não possui informações sobre o “PowerCharge”, impossibilitando-o de iniciar a recarga de seu veículo, dessa forma ele é forçado a instalar outro aplicativo indicado na estrutura física do eletroposto para realizar o carregamento. Isso destaca uma lacuna na comunicação entre aplicativos e eletropostos, revelando uma falta de interoperabilidade entre esses agentes.

Após essa situação desconfortável, Luiz entra em contato com o aplicativo “EcoCharge Solutions” e relata o ocorrido, a resposta da empresa é de que seria inviável gerenciar conexões com tantos eletropostos, por isso esse eletroposto não estava disponível no aplicativo. Os desenvolvedores do “EcoCharge Solutions” insatisfeitos com essa situação e visando uma oportunidade de mercado para expandir o seu número de usuários, decidem mudar a abordagem de conexão com eletropostos, pois foi entendido que a complexidade anterior de gerenciar conexões com cada eletroposto era inviável para o crescimento do negócio.

A Figura 1 ilustra a conexão do “EcoCharge Solutions” com os eletropostos, evidenciando que a comunicação ocorre apenas com o eletroposto “TechPower”.

Figura 1 - Abordagem atual de conexões da "EcoCharge Solutions”

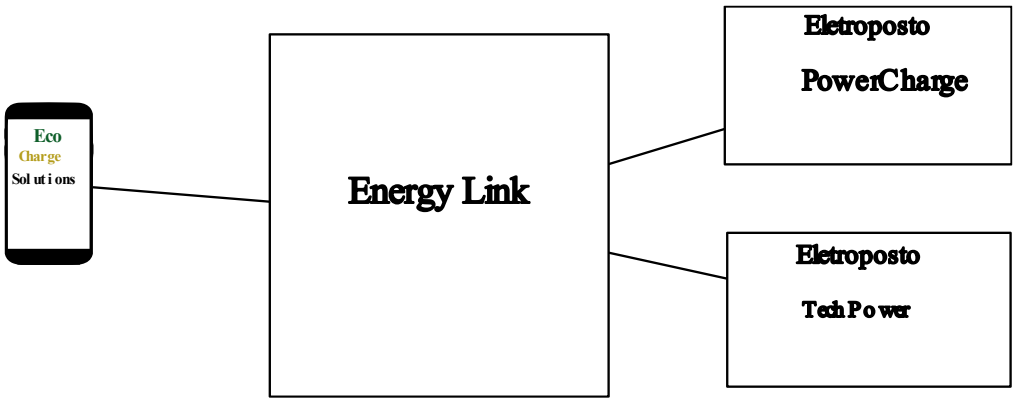


Fonte: Próprio Autor.

Imediatamente, os desenvolvedores encontram o “EnergyLink”, uma empresa que propõe uma solução para o gerenciamento de conexões entre aplicativos e eletropostos. A partir do estudo sobre o “EnergyLink”, foi visto que não existe mais conexões diretas entre aplicativos e eletropostos, mas sim conexões entre aplicativos e o “EnergyLink”, e eletropostos e “EnergyLink”. A partir desse entendimento o “EcoCharge Solutions” toma a decisão de se conectar com o “EnergyLink”, pois agora não precisará se conectar diretamente com cada eletroposto e ainda assim terá acesso a todos eletropostos que estão conectados no “EnergyLink”. É importante ressaltar que essa integração do “EcoCharge Solutions” com o “EnergyLink” não é apenas vantajosa para o aplicativo, pois com a entrada dele, os serviços dos eletropostos alcançarão mais pessoas. A Figura 2 mostra o estado atual das conexões

da “EchoCharge Solutions” agora se comunicando com 2 eletropostos através do “EnergyLink”.

Figura 2 - Nova abordagem de conexões da "EcoCharge Solutions"



Fonte: Próprio Autor.

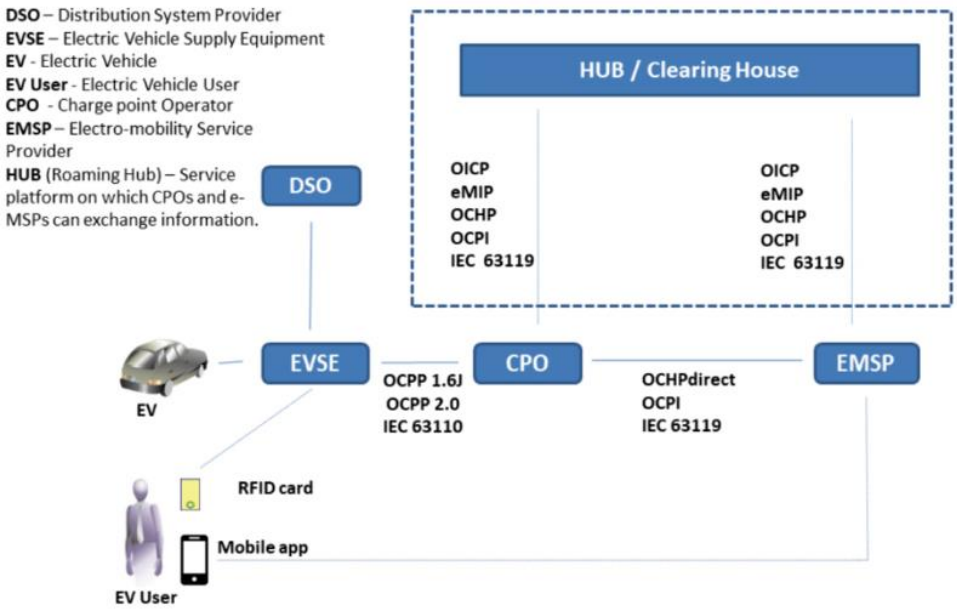
Uma semana depois, Luiz em sua viagem de volta para Santa Maria, necessita parar para carregar seu veículo no “PowerCharge”, após ler o anúncio de que o “EcoCharge Solutions” foi conectado com o “EnergyLink”, ele decide abrir o “EcoCharge Solutions” e para sua surpresa não apenas as informações sobre o “PowerCharge” estão disponíveis, mas como também sobre outros eletropostos até mesmo fora de seu país.

Nessa narrativa, o caso de Luiz destaca a importância crítica de uma comunicação eficaz entre serviços de carregamento. O “EcoCharge Solutions” é um eMSP, os eletropostos “TechPower” e “PowerCharge” são EVCSs e o “EnergyLink” é um roaming hub. Na história contada os EVCSs se comunicam diretamente com o roaming hub ou com o eMSP, porém na realidade quem faz a comunicação dos carregadores de um EVCS com o mundo externo é o CPO, através dessa informação é possível inferir que o “TechPower” e o “PowerCharge” são operados por CPOs distintos, é possível inferir também que o “PowerCharge” está conectado com o “EnergyLink”, pois após a integração do “EcoCharge Solutions” com o roaming hub, este eletroposto ficou disponível.

A partir da Figura 3, percebe-se que para a comunicação do usuário com o carregador através de um eMSP é necessário um protocolo, como por exemplo o

OCPI. O eMSP sempre se comunicará com o CPO, indiretamente através de um HUB ou diretamente, mas em ambas as formas um protocolo será necessário.

Figura 3 - Comunicação do usuário com o carregador



Fonte: (Electric Vehicle Recharging and Roaming Protocols: A Brazilian Electromobility Case Study, 2022).

A comunicação entre CPOs e eMSPs pode ser feita de forma direta (*peer-to-peer*), que é um CPO se comunicando diretamente com um eMSP, ou através de um intermediário que seria o *roaming hub*, que é o CPO se comunicando com o *roaming hub* e o *roaming hub* se comunicando com o eMSP.

Tanto a comunicação via *roaming hub* quanto a comunicação direta *peer-to-peer* possuem vantagens e desvantagens como contatadas no Quadro 1.

Quadro 1 – Peer-2-Peer e Roaming hub, vantagens e desvantagens da perspectiva de CPOs e eMSPs

	Peer-2-Peer	**Roaming hub**
Vantagens	Flexível e customizável: Possibilita a discussão e acordo de todos os aspectos técnicos e comerciais.	Permite acesso intermediário a todos que estão conectados no determinado *roaming hub.* Oferece um único, conjunto harmônico de acordos técnicos (protocolos, implementação, etc.). Implementa, mantém e atualiza os protocolos de roaming. Pode incluir uma estrutura única e harmônica de acordos comerciais de roaming, e pode gerenciar mudanças nesses acordos, devido a mudanças legislativas, de tarifas ou operacionais. Pode incluir serviços de teste de certificação/implementação. Pode permitir conexões que utilizam protocolos diferentes. Pode oferecer serviços administrativos como cobrança, faturamento, liquidação de pagamentos e compensação.
Desvantagens	Negociar muitas conexões de roaming individualmente pode ser custar tanto tempo quanto dinheiro Pode precisar de um custo significante de implementação técnica, especialmente se a implementação difere entre cada conexão individual.	Menor granularidade: *roaming hubs* decidem quais protocolos usar, e pode definir as regras de negócio. Menos possibilidade de customizar ou adaptar necessidades individuais. *Roaming hubs* irão cobrar uma taxa pelos seus serviços.

Fonte: (Bekkers e Van der Kam, 2020, tradução nossa)

É importante destacar que um dos principais objetivos de CPOs e eMSPs é possuir o maior número de usuários possível, Através de um roaming hub, tanto o CPO quanto o eMSP se beneficiarão, pois, por um lado, quanto mais eMSPs estiverem conectados com o CPO, mais usuários terão acesso a ele e consequentemente, mais do seu serviço ele irá vender, pelo outro lado, quanto mais EVCSs estiverem disponíveis para um eMSP, mais qualidade e serviço ele poderá oferecer a seus usuários. É válido ressaltar que a utilização de um roaming hub não implica necessariamente em um aumento direto de interoperabilidade. No entanto, essa estratégia torna o alcance desse objetivo mais tangível e passível de ser alcançado.

A escolha de utilização de um *roaming hub* ou uma conexão peer-2-peer vai variar de acordo com os recursos e objetivos de cada CPO/eMSP. CPOs/eMSPs que possuem menos recursos, optarão a utilização de *roaming hubs*, já para CPOs/eMSPs maiores e com mais recursos, peer-2-peer é uma opção mais atrativa, caso esses CPOs/EMSPs queiram se conectar com outros CPOs/EMSPs menores, terão que se conectar via um *roaming hub* (BEKKERS e VAN DER KAM, 2020).

2.3 OPEN CHARGE POINT INTERFACE (OCPI)

De forma sucinta, o OCPI é um protocolo que possibilita a comunicação entre CPOs e eMSPs através de uma linguagem uniforme. É um protocolo baseado em HTTP e utiliza JSON como forma de troca de dados.

O OCPI tem como funcionalidades principais:

- Um bom sistema de roaming (para uso bilateral ou via hub).
- Informação em tempo real sobre localização, disponibilidade e preço.
- Um jeito uniforme de troca de dados, antes, durante e após a transação.
- Suporte móvel para acessar qualquer carregador sem um cadastro prévio.

A seguir serão apresentados os seguintes módulos do protocolo OCPI versão 2.2: Versions, Credentials, Locations, Sessions, CDRs, Tariffs, Tokens, Commands, Charging Profiles, HubClientInfo.

2.3.1 Versions

Esse módulo do OCPI é fundamental para a implementação de uma conexão entre duas entidades, como, por exemplo, um CPO e um eMSP, pois disponibiliza informações sobre as versões suportadas e os detalhes sobre cada versão. Essas informações são essenciais para a estabilização de uma conexão inicial através da troca de tokens abordadas mais detalhadamente no capítulo 2.3.2.

Para um cliente conhecer os *endpoints* e as versões da implementação esse módulo é extremamente útil. O cliente pode requisitar as versões através do *endpoint* versions e o servidor deve retornar todas as versões suportadas como ilustrado na Figura 4.

Figura 4 - Lista de versões suportadas por uma determinada implementação do OCPI

```
[
  {
    "version": "2.1.1",
    "url": "https://www.server.com/ocpi/2.1.1/"
  },
  {
    "version": "2.2",
    "url": "https://www.server.com/ocpi/2.2/"
  }
]
```


Fonte - (OCPI, 2.2, 2020)

O campo “url” pode ser utilizado para obter detalhes de determinada versão, como os módulos que determinada versão possui e seus *endpoints*.

2.3.2 Credentials

Um pré-requisito para a utilização do OCPI é a troca de tokens. Esses tokens serão utilizados para autorizar o acesso a *endpoints*. Para a troca de tokens começar, é necessário que as partes decidam quem será o enviador e o receptor de credenciais.

O receptor deverá criar um token e enviar ao enviador, com esse token o enviador já pode acessar os *endpoints* do receptor. O enviador deve gerar um novo token e enviar ao receptor no módulo de credenciais, com esse token o receptor pode acessar os *endpoints* do enviador, como resposta dessa requisição, o receptor envia

um novo token que deve ser utilizado para todas as futuras requisições que o enviador fizer ao receptor.

O objeto de credenciais é representado como ilustrado na Figura 5.

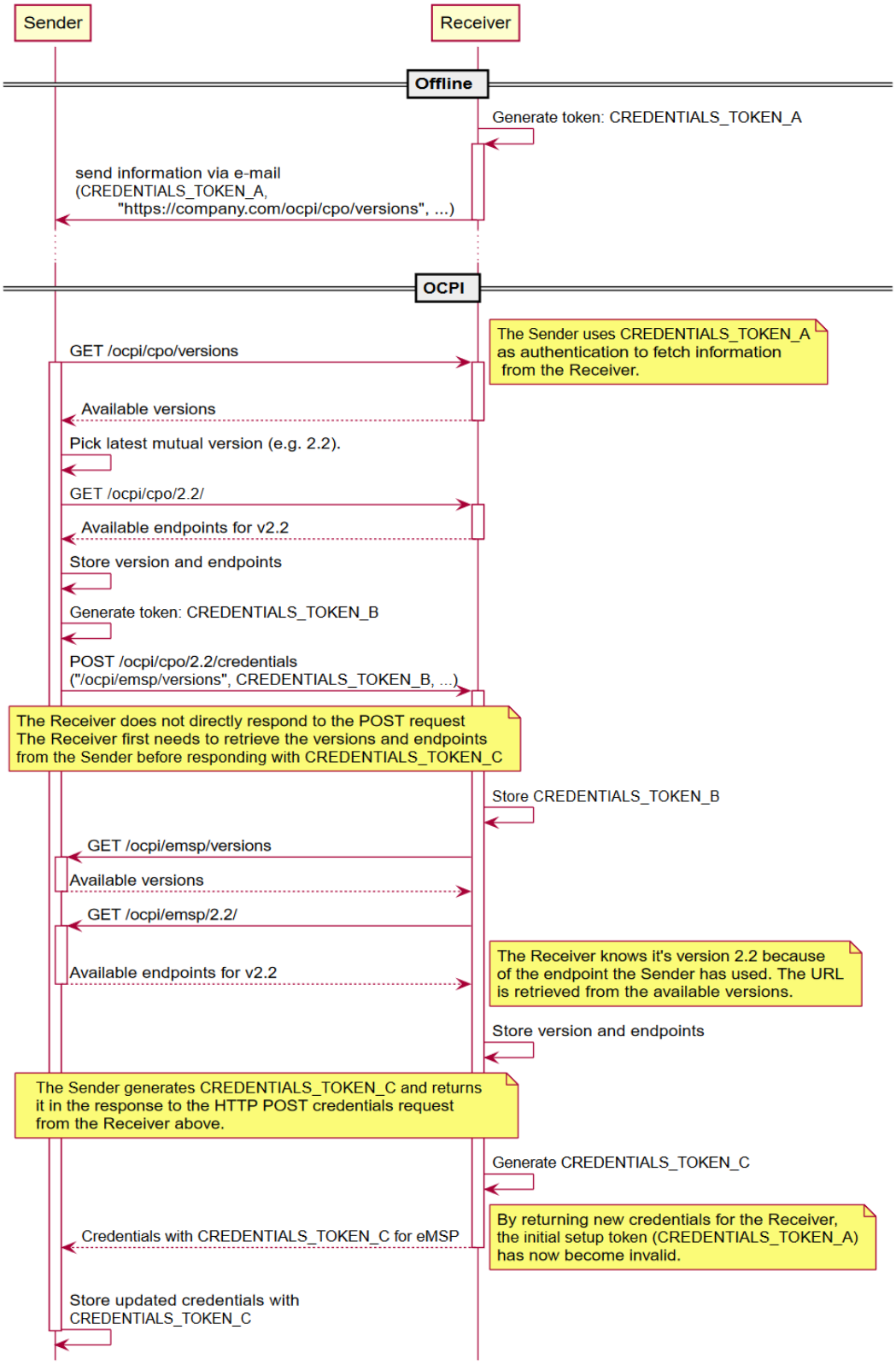
Figura 5 - Exemplo de um objeto de credenciais mínimo de um CPO

```
{
    "token": "ebf3b399-779f-4497-9b9d-ac6ad3cc44d2",
    "url": "https://example.com/ocpi/versions/",
    "roles": [{
        "role": "CPO",
        "party_id": "EXA",
        "country_code": "NL",
        "business_details": {
            "name": "Example Operator"
        }
    }]
}
```

Fonte - (OCPI, 2.2, 2020)

O processo de registro de credenciais do OCPI é ilustrado na Figura 6.

Figura 6 - Processo de registro OCPI



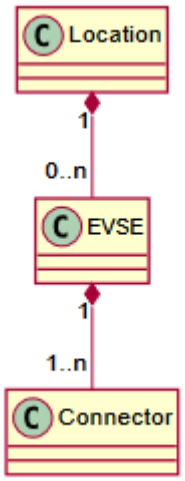
Fonte - (OCPI, 2.2, 2020)

A Figura 6 ilustra o funcionamento do processo de registro no OCPI, com o *Sender* sendo o enviador e o *Receiver* sendo o receptor. É importante notar que esse módulo é obrigatório em todas as implementações do OCPI, pois é através dele que os tokens necessários são obtidos para autenticação nas requisições. Como não existe um login com senha entre plataformas, o módulo de credenciais é o que faz essa função.

2.3.3 Locations

A localização é um objeto criado pelo CPO que descreve a localização de uma ou mais estações de carregamento como mostrado na Figura 7.

Figura 7 - Diagrama de classes da Localização



Fonte - (OCPI, 2.2, 2020)

Sendo uma das informações mais úteis para o usuário, é necessário que quando haja mudança, esta seja atualizada o mais breve possível para que os eMSPs tenham a versão atualizada. A Figura 8 mostra como um objeto de Localização é representado no OCPI.

“Quando uma versão não está disponível para carregamento público ou roaming, é recomendado não enviar esta localização via OCPI”, (OCPI, 2.2, 2020, tradução nossa).

Figura 8 - Exemplo de um objeto de Localização


```
{
  "country_code": "BE",
  "party_id": "BEC",
  "id": "LOC1",
  "publish": true,
  "name": "Gent Zuid",
  "address": "F.Rooseveltlaan 3A",
  "city": "Gent",
  "postal_code": "9000",
  "country": "BEL",
  "coordinates": {
    "latitude": "51.047599",
    "longitude": "3.729944"
  },
  "parking_type": "ON_STREET",
  "evses": [{
    "uid": "3256",
    "evse_id": "BE*BEC*E041503001",
    "status": "AVAILABLE",
    "status_schedule": [],
    "capabilities": [
      "RESERVABLE"
    ],
    "connectors": [{
      "id": "1",
      "standard": "IEC_62196_T2",
      "format": "CABLE",
      "power_type": "AC_3_PHASE",
      "voltage": 220,
      "amperage": 16,
      "tariff_ids": ["11"],
      "last_updated": "2015-03-16T10:10:02Z"
    }, {
      "id": "2",
      "standard": "IEC_62196_T2",
      "format": "SOCKET",
      "power_type": "AC_3_PHASE",
      "voltage": 220,
      "amperage": 16,
      "tariff_ids": ["13"],
      "last_updated": "2015-03-18T08:12:01Z"
    }],
    "physical_reference": "1",
    "floor_level": "-1",
    "last_updated": "2015-06-28T08:12:01Z"
  }, {
    "uid": "3257",
    "evse_id": "BE*BEC*E041503002",
    "status": "RESERVED",
    "capabilities": [
      "RESERVABLE"
    ],
    "connectors": [{
      "id": "1",
      "standard": "IEC_62196_T2",
      "format": "SOCKET",
      "power_type": "AC_3_PHASE",
      "voltage": 220,
      "amperage": 16,
      "tariff_ids": ["12"],
      "last_updated": "2015-06-29T20:39:09Z"
    }],
    "physical_reference": "2",
    "floor_level": "-2",
    "last_updated": "2015-06-29T20:39:09Z"
  }],
  "operator": {
    "name": "BeCharged"
  },
  "time_zone": "Europe/Brussels",
  "last_updated": "2015-06-29T20:39:09Z"
}
```


Fonte - (OCPI, 2.2, 2020)

O objeto de Localização é modelado de forma robusta, com muitos atributos para que o usuário tenha um grande número de informações sobre determinada localização.

2.3.4 Sessions

O módulo de sessões tem como escopo as sessões de carregamento, um objeto de sessão tem como objetivo descrever uma sessão de carregamento.

Quando o usuário inicia ou reserva uma sessão de carregamento, um objeto de sessão deve ser criado e armazenado no CPO. Para um EMSP ter as informações de sessões de carregamento de um determinado CPO, é necessário que o EMSP faça uma requisição do tipo GET para o CPO com a estrutura mostrada na Figura 9 a seguir.

Figura 9 - Estrutura de uma requisição do tipo GET para retornar as sessões

```
{sessions_endpoint_url}?[date_from={date_from}]&[date_to={date_to}]&[offset={offset}]&[limit={li
mit}]
```

Fonte - (OCPI, 2.2, 2020)

A sessão é importante para o usuário ter conhecimento do estado atual de carregamento do seu veículo. A figura 10 mostra os atributos de um objeto de sessão como por exemplo o id da localização, a data de início da sessão e a quantidade de energia consumida em kWh.

Figura 10 - Exemplo de um objeto de sessão simples


```
{
    "country_code": "NL",
    "party_id": "STK",
    "id": "101",
    "start_date_time": "2020-03-09T10:17:09Z",
    "kwh": 0.0,
    "cdr_token": {
        "uid": "123abc",
        "type": "RFID",
        "contract_id": "NL-TST-C12345678-S"
    },
    "auth_method": "WHITELIST",
    "location_id": "LOC1",
    "evse_uid": "3256",
    "connector_id": "1",
    "currency": "EUR",
    "total_cost": {
        "excl_vat": 2.5
    },
    "status": "PENDING",
    "last_updated": "2020-03-09T10:17:09Z"
}
```


Fonte - (OCPI, 2.2, 2020)

2.3.5 CDRs

Um Registro de Dados de Carregamento (CDR) constitui a descrição de uma sessão de carregamento concluída. Quando um carregamento é realizado, o CPO deve enviar ao eMSP um CDR, contendo informações sobre este determinado carregamento, principalmente informações sobre custo, conforme pode ser verificado na Figura 11. Isso permite que o usuário compreenda os fatores que contribuíram para o cálculo do custo total, proporcionando transparência e clareza sobre os componentes envolvidos no custo associado à sessão de carregamento.

Figura 11 - Exemplo de um CDR


```
{
  "country_code": "BE",
  "party_id": "BEC",
  "id": "12345",
  "start_date_time": "2015-06-29T21:39:09Z",
  "end_date_time": "2015-06-29T23:37:32Z",
  "cdr_token": {
    "uid": "012345678",
    "type": "RFID",
    "contract_id": "DE8ACC12E46L89"
  },
  "auth_method": "WHITELIST",
  "cdr_location": {
    "id": "LOC1",
    "name": "Gent Zuid",
    "address": "F.Rooseveltlaan 3A",
    "city": "Gent",
    "postal_code": "9000",
    "country": "BEL",
    "coordinates": {
      "latitude": "3.729944",
      "longitude": "51.047599"
    },
    "evses_uid": "3256",
    "evse_id": "BE*BEC*E041503003",
    "connectors_id": "1",
    "connectors_standard": "IEC_62196_T2",
    "connectors_format": "SOCKET",
    "connectors_power_type": "AC_1_PHASE"
  },
  "currency": "EUR",
  "tariffs": [{
    "country_code": "BE",
    "party_id": "BEC",
    "id": "12",
    "currency": "EUR",
    "elements": [{
      "price_components": [{
        "type": "TIME",
        "price": 2.00,
        "vat": 10.0,
        "step_size": 300
      }]
    }],
    "last_updated": "2015-02-02T14:15:01Z"
  }],
  "charging_periods": [{
    "start_date_time": "2015-06-29T21:39:09Z",
    "dimensions": [{
      "type": "TIME",
      "volume": 1.973
    }],
    "tariff_id": "12"
  }],
  "total_cost": {
    "excl_vat": 4.00,
    "incl_vat": 4.40
  },
  "total_energy": 15.342,
  "total_time": 1.973,
  "total_time_cost": {
    "excl_vat": 4.00,
    "incl_vat": 4.40
  },
  "last_updated": "2015-06-29T22:01:13Z"
}
```


Fonte - (OCPI, 2.2, 2020)

O CDR possui diversos atributos, através desse objeto é possível identificar as informações de uma sessão de carregamento encerrada.

2.3.6 Tariffs

Através do módulo de tarifas, os eMSPs conhecem as tarifas de CPOs, para isso os CPOs necessitam enviar as tarifas para os eMSPs.

Os CPOs definem suas tarifas da forma necessária, uma tarifa pode ficar complexa dependendo do que o CPO definir para a cobrança. Basicamente uma tarifa é a forma que um CPO realiza a cobrança pela utilização de sua infraestrutura física de carregamento. A Figura 12 ilustra um objeto de uma tarifa simples.

Figura 12 - Exemplo de uma tarifa simples


```
{
  "country_code": "DE",
  "party_id": "ALL",
  "id": "16",
  "currency": "EUR",
  "elements": [{
    "price_components": [{
      "type": "ENERGY",
      "price": 0.25,
      "vat": 10.0,
      "step_size": 1
    }]
  }],
  "last_updated": "2018-12-17T11:16:55Z"
}
```


Fonte - (OCPI, 2.2, 2020)

O cálculo do valor da tarifa da Figura 12 é feito da seguinte forma, levando em conta que 0.25 euros serão cobrados por kWh e 20kWh foram carregados:

$$0.25 \times 20 = 5\ (sem\ VAT) \quad (1)$$

$$(0.25 \times 20) + (10 \times (0.25 \times 20) \div 100) = 5.5\ (com\ VAT)$$

2.3.7 Tokens

Esse módulo possibilita o armazenamento de tokens de eMSPs por CPOs. Dessa forma o CPO pode armazenar esse token e autorizar o carregamento. Um objeto do tipo Token é representado no OCPI como mostra a Figura 13.

Figura 13 - Exemplo de um objeto do tipo Token


```
{
  "country_code": "DE",
  "party_id": "TNM",
  "uid": "bdf21bce-fc97-11e8-8eb2-f2801f1b9fd1",
  "type": "APP_USER",
  "contract_id": "DE8ACC12E46L89",
  "issuer": "TheNewMotion",
  "valid": true,
  "whitelist": "ALLOWED",
  "last_updated": "2018-12-10T17:16:15Z"
}
```


Fonte - (OCPI, 2.2, 2020)

Os tokens podem ser utilizados para autorização em tempo real, para isso é necessário que o atributo “whitelist” do objeto Token tenha o valor “NEVER” e o eMSP tenha o endpoint de autorização implementado. Com a autorização em tempo real, o CPO ao invés de autorizar o token armazenado em seu próprio sistema, agora necessita fazer uma requisição ao eMSP para a autorização.

De acordo com o OCPI, autorização em tempo real pode gerar um atraso maior, dessa maneira, gerando uma experiência ruim para o usuário que quer carregar o seu veículo, então é necessário cuidado na decisão de escolha sobre o tipo de autorização a ser realizada.

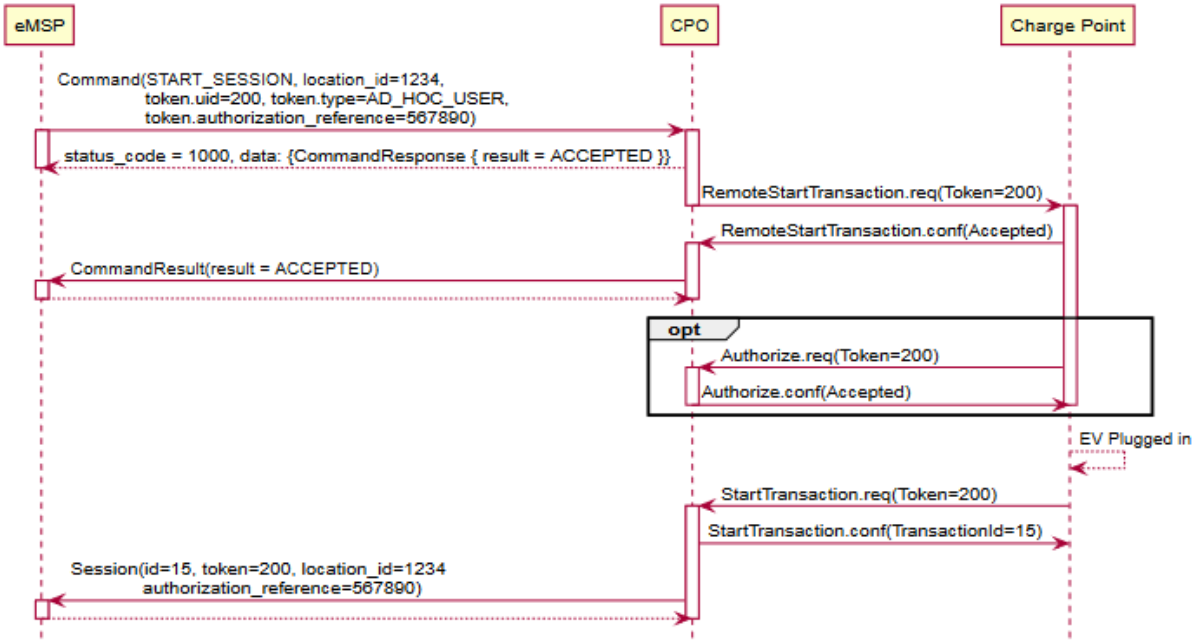
2.3.8 Commands

Esse módulo permite que comandos sejam executados de forma remota, por exemplo, através de um eMSP.

Os comandos são enviados geralmente de um eMSP para um CPO, quando o CPO recebe a requisição ele se comunica com o carregador solicitando a execução do comando desejado e retorna para o eMSP a resposta de acordo com o resultado

do carregador. A Figura 14 apresenta o fluxo de uma requisição de um eMSP, através de comando remoto, realizada com sucesso.

Figura 14 - Fluxo de requisição de comando remoto



Fonte - (OCPI, 2.2, 2020)

Os commandos remotos suportados, conforme OCPI 2.2 (2020) são: RESERVE_NOW, CANCEL_RESERVATION, START_SESSION, STOP_SESSION, UNLOCK_CONNECTOR. Cada comando tem a funcionalidade associado ao seu nome em inglês.

RESERVE_NOW significa “reservar agora” e permite que reservas de conectores sejam realizadas de forma remota.

CANCEL_RESERVATION significa “cancelar reserva” e permite que uma reserva feita seja cancelada remotamente.

START_SESSION significa “iniciar sessão” e permite que uma sessão de carregamento seja iniciada remotamente.

STOP_SESSION significa “parar sessão” e permite que uma sessão em andamento seja interrompida remotamente.

UNLOCK_CONNECTOR significa “desbloquear conector” e permite que o conector seja desbloqueado do veículo remotamente. Essa funcionalidade pode ser útil quando alguma falha ocorre e o conector conectado ao veículo elétrico não sai.

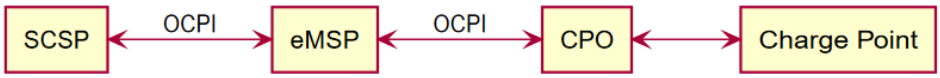
2.3.9 Charging Profiles

O módulo de Perfis de Carregamento permite um controle refinado sobre o processo de carregamento, possibilitando a otimização, flexibilidade e coordenação na gestão do carregamento de veículos elétricos. Ele desempenha um papel crucial em garantir operações eficientes e sustentáveis de carregamento ao alinhar o comportamento de carregamento com diversos fatores, incluindo as condições da rede elétrica, a disponibilidade da fonte de energia e as preferências do usuário.

“O carregamento inteligente usa inteligência para definir como um veículo elétrico irá receber eletricidade baseado no custo, na disponibilidade e nas necessidades do usuário. O carregamento inteligente possibilita a monitoração, o gerenciamento e o ajuste de consumo de energia”, (Driivz Team, 2022, tradução nossa).

Serviços que disponibilizam o carregamento inteligente por meio de algoritmos, comunicação em tempo real e integração com a rede elétrica são os *Smart Charging Services Providers (SCSP)*. Os SCSP podem ser utilizados no OCPI para a criação de Perfis de Carregamento como ilustrado na Figura 15.

Figura 15 - Topologia de carregamento inteligente em que o SCSP gera um Perfil de Carregamento

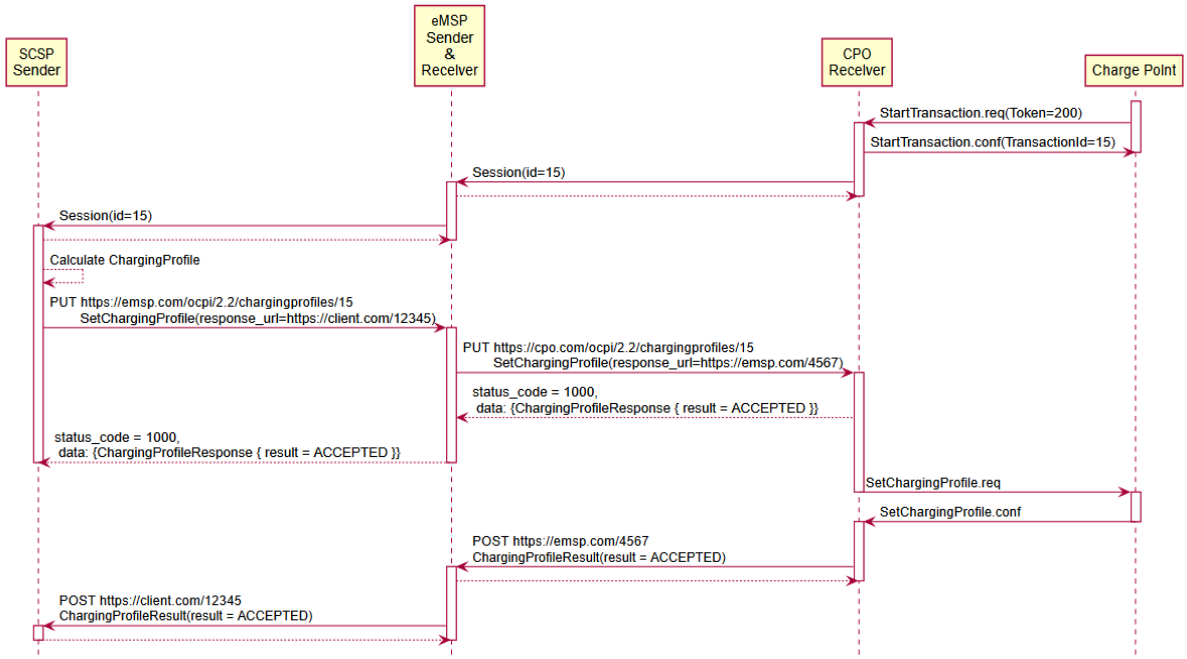


Fonte - (OCPI, 2.2, 2020)

A Figura 15 ilustra as comunicações para a criação de um Perfil de Carregamento por um SCSP, o eMSP delega a criação enviando objetos de Sessão para o SCSP que por sua vez retorna Perfis de Carregamento baseado nos objetos de Sessão recebidos, o eMSP envia para o CPO esses Perfis de Carregamento.

A Figura 16 ilustra o fluxo para definição de um Perfil de Carregamento que começa quando uma nova sessão é iniciada ou uma sessão existente é atualizada.

Figura 16 - Fluxo para definição de um Perfil de Carregamento



Fonte - (OCPI, 2.2, 2020)

O CPO enviará o objeto de Sessão para o eMSP que, nesse caso, está delegando a função de criação do Perfil de Carregamento para um SCSP, então o eMSP envia o objeto de Sessão para o SCSP que calcula um Perfil de Carregamento e, através do método HTTP PUT, envia o Perfil de Carregamento para o eMSP que envia para o CPO, que responde a requisição informando se pode ser enviada ao *Charge Point (Carregador)*, essa resposta é enviada ao SCSP pelo eMSP. O CPO envia a requisição para o carregador e quando recebe uma resposta do carregador envia para o eMSP utilizando o método HTTP POST, agora o eMSP envia a resposta para o SCSP utilizando o método HTTP PUT. Essa resposta terá um objeto *ChargingProfileResult* que, por sua vez, contém um *ChargingProfileResultType,* que está ilustrado na Figura 17.

Figura 17 - ChargingProfileResultType enum

Value	Description
ACCEPTED	ChargingProfile request accepted by the EVSE.
REJECTED	ChargingProfile request rejected by the EVSE.
UNKNOWN	No Charging Profile(s) were found by the EVSE matching the request.

Fonte - (OCPI, 2.2, 2020)

2.3.10 HubClientInfo

Este módulo é aplicável para CPOs e eMSPs que tenham sua comunicação executada através de um Hub. O módulo tem como objetivo fornecer informação sobre o status de cada cliente do Hub conforme a Figura 18. Desse modo, CPOs e eMSPs sabem quais estão conectados para comunicação, “Diferente de outros módulos do OCPI, esse módulo é entre eMSP/CPO e Hub ao invés de eMSP e CPO”, (OCPI, 2.2, “tradução nossa”).

Figura 18 - ConnectionStatus enum

Value	Description
CONNECTED	Party is connected.
OFFLINE	Party is currently not connected.
PLANNED	Connection to this party is planned, but has never been connected.
SUSPENDED	Party is now longer active, will never connect anymore.

Fonte - (OCPI, 2.2, 2020)

"O Hub precisa determinar se uma conexão ainda está viva. Para fazer isso, o Hub deve ter controle do tempo que passou desde a última mensagem recebida por uma parte conectada. Quando o tempo é maior que X minutos (quando em dúvida, começar com 5 minutos) o Hub deve enviar uma: GET para o *endpoint* de informação de versão. Pois esse *endpoint* é obrigatório no OCPI e é providenciado por todas as partes, e um GET para o *endpoin*t de versões não causa nenhum efeito colateral, isso é visto como o melhor modo de se fazer uma checagem de ‘ainda-vivo’.”, (OCPI, 2.2,

2020, tradução nossa). Um objeto do tipo ClientInfo é representado da forma mostrada na Figura 19.

Figura 19 - Objeto ClientInfo

Property	Type	Card.	Description
party_id	CiString(3)	1	CPO or eMSP ID of this party (following the 15118 ISO standard), as used in the credentials exchange.
country_code	CiString(2)	1	Country code of the country this party is operating in, as used in the credentials exchange.
role	Role	1	The role of the connected party.
status	ConnectionStatus	1	Status of the connection to the party.
last_updated	DateTime	1	Timestamp when this ClientInfo object was last updated.

Fonte - (OCPI, 2.2, 2020)

2.4 SIMULAÇÃO DE UM CARREGAMENTO UTILIZANDO O OCPI

O diagrama da Figura 20 é uma simulação de um carregamento feito por um usuário de um eMSP realizando a comunicação com um CPO através do protocolo OCPI. O diagrama tem como objetivo ilustrar o fluxo de requisições desde o momento da primeira conexão entre o eMSP e o CPO até após o encerramento da sessão de carregamento.

Por questões de simplificação, as requisições não estão formatadas de acordo com as especificações necessárias para o funcionamento em um ambiente de produção real.

Além disso, é importante ressaltar que os dados retornados como resposta pelas requisições representados neste diagrama não incluem todos os dados necessários para o funcionamento em um ambiente de produção real. Por motivos de simplificação e legibilidade, alguns dados foram omitidos.

Figura 20 - Requisições OCPI simplificadas

Fonte: Próprio Autor

As representações gráficas das setas utilizadas no diagrama da Figura 20 indicam informações cruciais sobre o fluxo de comunicação no contexto do OCPI. É fundamental compreender a simbologia empregada.

- Setas pretas: Estas setas simbolizam requisições, ou seja, ações iniciadas por uma entidade que solicita informações ou a realização de uma tarefa específica no sistema.
- Setas verdes: Representam as respostas às requisições anteriormente mencionadas. Essas respostas contêm os resultados ou informações solicitadas, demonstrando o funcionamento da comunicação no sistema.

- Seta cinza: Simboliza comunicação fora do OCPI. Esta seta representa a troca de informações ou ações que ocorrem fora do âmbito do protocolo OCPI, mas que são relevantes para o contexto geral do processo.

O primeiro passo no estabelecimento de uma conexão no OCPI envolve a troca de tokens, que posteriormente serão utilizados como mecanismos de autenticação para requisições. A primeira seta, representando o compartilhamento do “TOKEN_A” e do *endpoint* de versões do CPO, ocorre através de um meio de comunicação que não está dentro do escopo do OCPI. As etapas subsequentes, que compreendem as setas até a décima terceira, referem-se à troca de credenciais, cujos detalhes são explicados de forma mais aprofundada no capítulo 2.3.2 deste trabalho.

Neste cenário simulado, consideramos um usuário que, por meio de um eMSP, visualiza as estações de carregamento suportadas por um CPO e realiza a reserva de uma sessão de carregamento. Quando o horário designado para a reserva chega, o usuário conecta o conector ao seu veículo, iniciando assim o processo de recarga.

Após o término da recarga, os custos associados são exibidos na tela, fornecendo informações relevantes ao usuário. Esse fluxo de eventos ilustra como o OCPI padroniza a interação entre os diferentes atores e sistemas no contexto da mobilidade elétrica.

2.5 OCPI NO MUNDO REAL

A versão abordada do OCPI neste trabalho é a versão 2.2.1, que é atualmente a versão oficial, (EVRoaming Foundation, 2024). Em março de 2024 foi publicado o módulo “*DirectPayment Solution”*, que pode ser utilizado como um adicional a versão 2.2.1 do OCPI, (EVRoaming Foundation, 2024). A versão 3.0 do OCPI está em desenvolvimento. “Além de algumas melhorias de arquitetura, também contém novos módulos/serviços aperfeiçoados.”, (EvRoaming Foundation, 2024, tradução nossa).

Os próximos subcapítulos abordam casos em que o OCPI foi implementado.

2.5.1 Freshmile x Chargemap

A *Freshmile* e a *Chargemap*, sendo respectivamente o maior CPO e o maior eMSP da França, adotaram o OCPI como protocolo de comunicação de seus serviços.

A comunicação via OCPI entre essas duas empresas após dezembro de 2020, que foi o período de lançamento da conexão OCPI, tornou-se mais fácil de gerenciar,

pois quando estações de carregamento da *Freshmile* são implantadas, elas estarão instantaneamente disponíveis para os usuários da *Chargemap*, (EVRoaming, 2021).

2.5.2 Serviço de Carregamento Inteligente

O OCPI foi utilizado como solução de interoperabilidade para o desenvolvimento de um serviço de carregamento inteligente independente. Através da utilização dos módulos necessários foi possível fazer um serviço que se comunica com o CPO para otimizar a energia durante uma sessão de carregamento.

Durante uma sessão de carregamento, o CPO se comunica com o carregador através de algum protocolo, geralmente o OCPP, então através desse protocolo o carregador envia os dados de consumo de energia para o CPO, que então envia para o SCSP (*Smart Charging Service Provider*) que é o serviço de carregamento inteligente, a partir desses dados, o SCSP cria perfis de carregamento através do módulo “*ChargingProfiles*”, abordado no capítulo 2.3.9 deste trabalho, e então envia os perfis de carregamento para o CPO que envia para o carregador e assim muda a forma com que a energia é gerenciada no carregamento, (Guillemin, S. et al., 2024).

De acordo com Safaya (2024), o OCPI não é o protocolo ideal para a integração com SCSPs, pois há apenas o suporte para o gerenciamento de energia para sessões de carregamento que estão em andamento, o que limita o número de operações que o OCPP pode executar, consequentemente abastecendo o serviço de carregamento inteligente com menos detalhamento.

Dessa forma, a partir da utilização do OCPI na implementação de um SCSP, o propósito da interoperabilidade, para o suporte a diferentes serviços e independência do sistema foi cumprido, por outro lado, a partir da visão de Safaya, para uma otimização mais refinada, é necessário um SCSP mais robusto que suporte outras operações OCPP que não estão no escopo do OCPI.

2.5.3 Scheidt & Bachmann x Chargepoint

A colaboração entre essas duas empresas que fornecem serviços de carregamento ocorreu com a utilização do OCPI após o novo Regulamento de Infraestrutura para Combustíveis Alternativos, (UNIÃO EUROPEIA, 2024).

“Em alinhamento com esse objetivo, a União Europeia visa acelerar a adoção de veículos elétricos através da remoção de obstáculos e fornecendo uma infraestrutura de carregamento uniforme, amigável ao usuário por todo o continente. Para colaborar com essa iniciativa, Scheid & Bachmann fechou parceria com ChargePoint para desenvolver uma solução avançada que facilita pagamentos diretos em carregamentos de veículos elétricos, independente da marca da estação de carregamento ou modelo.” (Energy-Retail-Solutions, 2024, tradução nossa).

A parceria entre essas duas empresas, com o uso do protocolo OCPI, representa um avanço significativo na criação de uma infraestrutura de carregamento de veículos elétricos mais acessível. Essa colaboração não apenas atende aos novos regulamentos da União Europeia, mas também colabora para a fomentação de futuras adoções da interoperabilidade em infraestruturas de carregamento.

3 IMPLEMENTAÇÃO DO MÓDULO DE CREDENCIAIS

Este capítulo será dividido em 6 seções, cada uma correspondendo a uma ou mais classes relevantes para o módulo, detalhando sua importância e funcionalidade. É importante notar que será abordado com um maior detalhamento o registro entre plataformas, tendo em vista que esta é a funcionalidade que permite a autenticação e é passo fundamental para qualquer implementação do OCPI. Não necessariamente será feita a abordagem de todos os atributos das classes, pois tornaria o capítulo muito extenso e com informações desnecessárias para o escopo do projeto.

A implementação do módulo de credenciais foi dividida em etapas que vão desde a criação das classes de modelo até a implementação dos *endpoints* específicos do módulo. A linguagem utilizada foi Java juntamente com a utilização do *framework Spring Boot*. A escolha dessas tecnologias deu-se ao fato de o OCPI ser puramente a padronização de objetos e APIS, com essas ferramentas, o processo de desenvolvimento tornou-se ágil devido ao suporte a estas funcionalidades.

Inicialmente, foram implementadas todas as classes de modelo presentes no OCPI. Este passo foi essencial para garantir uma base sólida e reutilizável para todas as implementações específicas subsequentes. As classes foram desenvolvidas conforme as especificações detalhadas no documento oficial do OCPI.

3.1 CREDENTIALS

Esta classe representa o modelo de uma Credencial, esse é o objeto central deste módulo. A Figura 21 ilustra como a classe foi implementada.

Figura 21 - Classe Credentials

```java
@JsonNaming(PropertyNamingStrategies.SnakeCaseStrategy.class)
@Data
@AllArgsConstructor
@NoArgsConstructor
@Builder
public class Credentials {
    private String token;
    private URL url;
    private List<CredentialsRole> roles;
}
```

Fonte: Próprio Autor

A classe possui algumas anotações de bibliotecas do *Spring Boot*. Um dos problemas encontrados no começo do desenvolvimento foi a formatação dos atributos do JSON, já que o OCPI tem a sua comunicação através desse formato a documentação estabelece que todos os atributos devem ser minúsculos e separados por traço inferior, porém em Java, o padrão adotado para nomear as variáveis não é esse, isso resultava em JSON com atributos diferentes da especificação do OCPI.

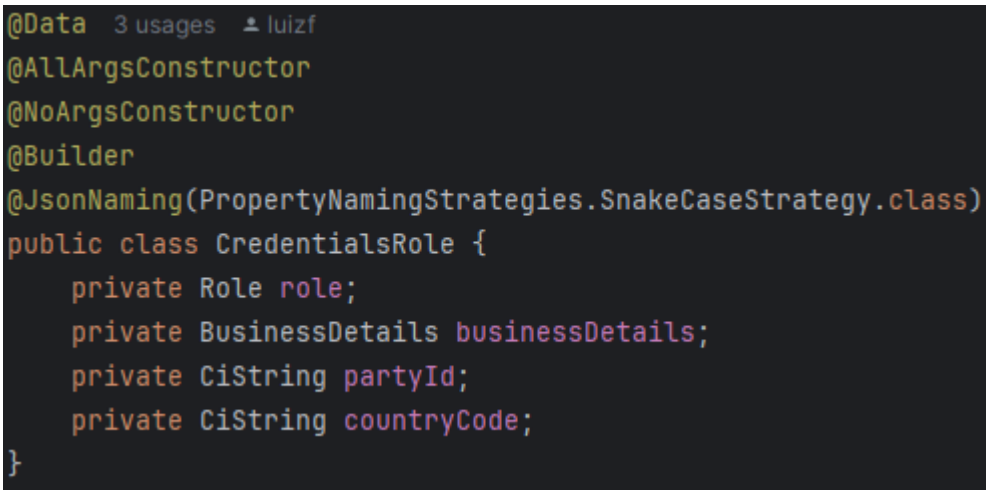
A primeira anotação é de uma biblioteca chamada *Jackson* que possui anotações para lidar com a serialização de objetos para JSON, e essa anotação com sua determinada especificação indica para o *Jackson* formatar o JSON com a mesma estratégia determinada no OCPI. A utilização dessa biblioteca, para serialização dos objetos, foi um fator crucial para a agilidade do desenvolvimento. O restante das anotações faz parte da biblioteca *Lombok*, que facilitam a legibilidade do código através da omissão de códigos repetitivos.

O atributo “token” refere-se ao token que será utilizado na autenticação entre as plataformas, uma plataforma necessita possuir um token válido para acessar os *endpoints* de outra. O atributo “*url*” refere-se ao *endpoint* do módulo de Versão dessa aplicação, portanto é necessário implementar esse módulo para prosseguir com o desenvolvimento do módulo de credenciais, através deste atributo a plataforma adquire os *endpoints* de outra e pode fazer requisições. O atributo “*roles*” refere-se a funções e informações de cada plataforma.

3.2 CREDENTIALS ROLE

Esta classe está ilustrada na Figura 22 e contém informações sobre determinado serviço.

Figura 22 - Classe CredentialsRole



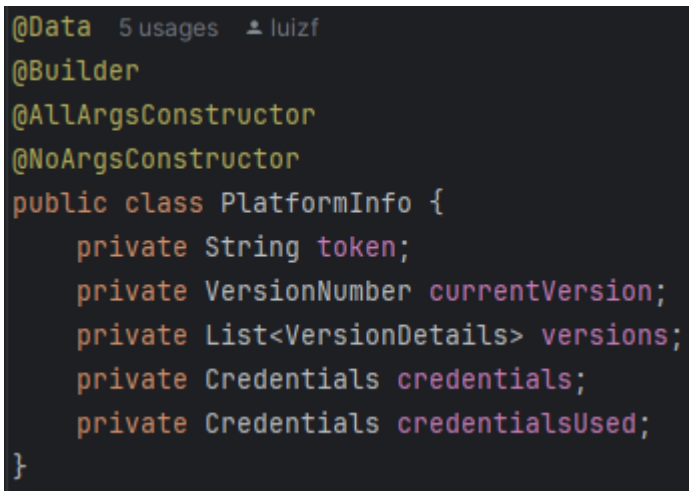
Fonte: Próprio Autor

O atributo “*role*” identifica o serviço como sendo: CPO, EMSP, HUB, NAP, NSP, SCSP e OTHER. A última identificação listada refere-se a outro serviço que não seja os anteriores. O atributo “*partyId*” é um identificador único do serviço, já o “*countryCode*” é um identificador único de País. Não é permitido serviços com um “*countryCode*” igual possuírem o mesmo “*partyId*”. A garantia da atribuição única de *partyIds* não está no escopo deste trabalho. O atributo “*businessDetails*” são as informações do nome, website e logo do serviço.

3.3 PLATFORM INFO

Esta classe não está especificada no OCPI, portanto é uma classe customizada. O protocolo não determina como os dados serão armazenados em cada aplicação, a classe foi criada para suprir a necessidade de armazenamento de dados relevantes de cada plataforma registrada e foi criada como mostra a Figura 23.

Figura 23 - Classe PlatformInfo



Fonte: Próprio Autor

O atributo “token” é o token necessário para autenticação com a plataforma. O atributo “*currentVersion*” é a versão que a comunicação ocorre atualmente entre as plataformas. O atributo “*versions*” possui detalhes sobre todas as versões suportadas dessa plataforma. O atributo “*credentials*” é o objeto de credenciais da plataforma, já o “*credentialsUsed*” é o objeto de *credentials* que essa plataforma utiliza para se autenticar.

3.4 CREDENTIALS TOKEN AUTH FILTER E CREDENTIALS TOKEN SERVICE

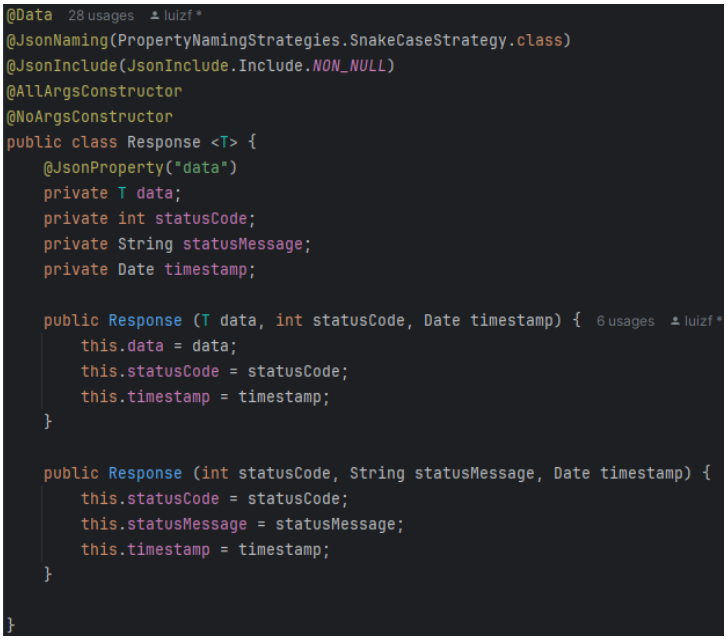
O OCPI determina que todas as requisições necessitam de autenticação com token. A classe “*CredentialsTokenAuthFilter*” tem como propósito interceptar todas as requisições para a aplicação e checar a validade do token para que a requisição prossiga. Esta classe atua como um guarda de um estabelecimento que controla a entrada de pessoas, vamos supor que para entrar neste estabelecimento seja necessário a identificação através de sua carteira de identidade, caso o guarda cheque a identidade e ela é válida, a pessoa poderá prosseguir para o interior do estabelecimento, caso contrário ela não poderá entrar.

Nessa analogia, a aplicação é o estabelecimento e a classe “*CredentialsTokenAuthFilter*” é o guarda. O modo de validação e outros métodos para a manipulação do token são implementados na classe “*CredentialsTokenService*”. Portanto o guarda da aplicação utiliza o “*CredentialsTokenService*” para validar o token. Este token é enviado no cabeçalho da requisição no campo “*Authorization*”.

3.5 CREDENTIALS CONTROLLER

Nesta classe são determinados os *endpoints* da aplicação, quando uma requisição passa pelo filtro, que é o “*CredentialsTokenAuthFilter*”, ela é destinada ao *endpoint* específico requisitado. O OCPI determina que a resposta de todas as requisições tenha o mesmo tipo, e esse tipo é o “*Response*”, não há o estabelecimento de um nome para essa classe, a escolha do nome foi feita devido a padronização linguística do protocolo que é a língua inglesa.

Figura 24 - Classe Response



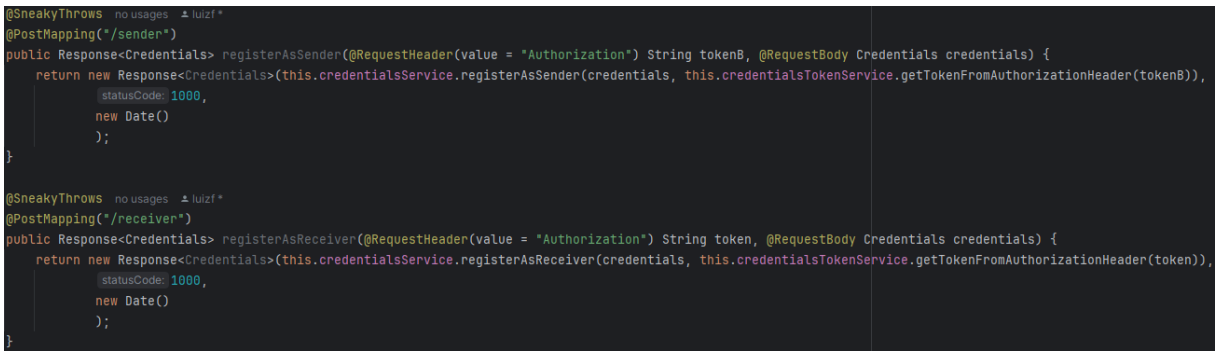
Fonte: Próprio Autor

Há quatro atributos nesta classe, “*data*” é o que se refere ao objeto que a aplicação quer enviar, por exemplo um objeto da classe “*Credentials*”. Este campo pode ter qualquer tipo, por isso ele foi definido como um campo de tipo genérico.

O atributo “*statusCode*” é o código de status que tem seu uso especificado pelo OCPI, “*statusMessage*” é a mensagem para ajudar em questões de depuração e “*timestamp*” é o horário e a data que o objeto foi criado.

A Figura 25 exibe a implementação de alguns dos *endpoints* do módulo de credenciais.

Figura 25 - Métodos de Registro da Classe CredentialsController



Fonte: Próprio Autor

Há dois métodos ilustrados na Figura 25, ambos são da funcionalidade de registro dentro do OCPI. O registro de credenciais no OCPI possui dois lados: O do enviador e o do receptor. O primeiro método "*registerAsSender*" faz o atua no registro como o enviador, já o método "*registerAsRreceiver*" atua como receptor, ambos retornam o tipo "*Response*" com o tipo "*data*" sendo uma "*Credentials*", que é retornada através dos métodos "*registerAsSender*" e "*registerAsReceiver*" da classe "*CredentialsService*".

Os dois métodos capturam o token do cabeçalho para enviar como parâmetro para as respectivas funções que lidam com a lógica do registro, juntamente com um objeto "*Credentials*" no corpo da requisição.

3.6 CREDENTIALS SERVICE

Esta classe é responsável pela lógica do módulo de credenciais, é nela que o processo ilustrado na Figura 6 é implementado. Para explicar a implementação do processo de registro este subcapítulo será dividido em duas partes, uma da implementação do enviador e a outra do receptor.

3.6.1 SENDER

O papel do enviador ou "*Sender*" é começar o processo dentro do OCPI. De acordo com o processo de registro, que está ilustrado na figura 6, o enviador de modo geral necessita cumprir os seguintes passos:

1. Obter as versões disponíveis do receptor
2. Obter os *endpoints* da versão mútua mais atualizada

3. Enviar um token válido para o receptor no módulo de credenciais
4. Armazenar o token retornado pelo receptor para requisições futuras

Com essa estrutura definida o próximo passo é investigar a implementação exibida na Figura 26.

Figura 26 - Método "registerAsSender"

```
public Credentials registerAsSender(final Credentials credentials, final String tokenB) throws PlatformAlreadyRegistered, NoMutualVersion, JsonProcessingException {
    final var credentialsEndpointOpt = getCredentialsEndpoint(credentials, tokenB, InterfaceRole.RECEIVER);
    Credentials finalCredentials = null;
    final var platform = this.platformData.getPlatforms().get(tokenB);
    if(credentialsEndpointOpt.isPresent()) {
        final var credentialsEndpoint = credentialsEndpointOpt.get();
        this.credentialsTokenService.validateToken(tokenB);
        finalCredentials = senderLogic(credentialsEndpoint, tokenB);
    }
    System.out.println(platform);
    var encodedToken = this.credentialsTokenService.encodeToken(platform.getToken());
    System.out.println("Token codificado: " + encodedToken);
    platform.setCredentialsUsed(finalCredentials);
    platform.setCredentials(finalCredentials);
    return finalCredentials;
}
```

Fonte: Próprio Autor

O método possui dois parâmetros, um objeto de “*Credentials*” e um token, o retorno é um objeto de “*Credentials*” e ele pode lançar exceções.

A primeira linha dentro do método chama a função “*getCredentialsEndpoint*” que está ilustrada na Figura 27.

Figura 27 - Função "getCredentialsEndpoint"

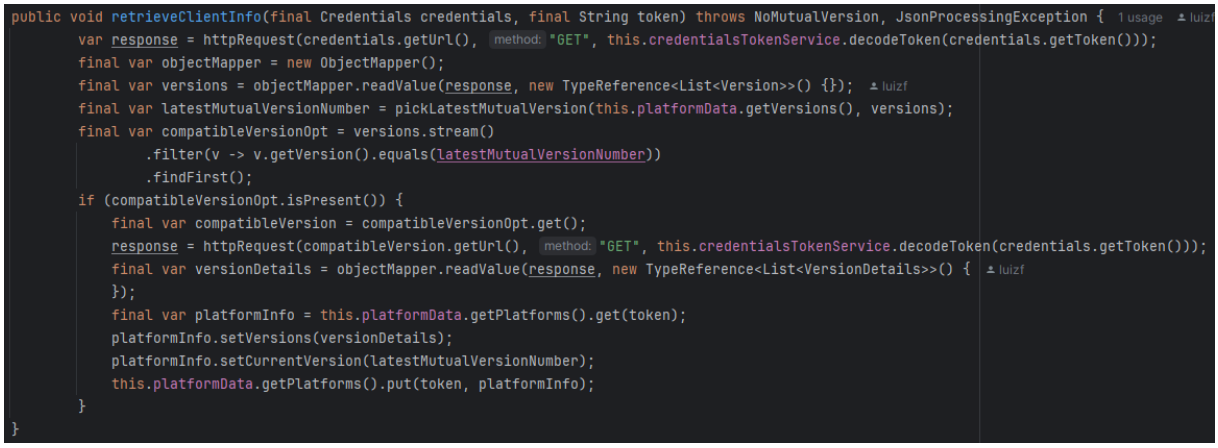
```
public Optional<Endpoint> getCredentialsEndpoint(final Credentials credentials, final String token, final InterfaceRole otherPlatformRole)
    if(!this.platformData.getPlatforms().containsKey(token)) {
        final var platformInfo = PlatformInfo.builder()
                .token(this.credentialsTokenService.decodeToken(credentials.getToken()))
                .build();
        this.platformData.getPlatforms().put(token, platformInfo);
        retrieveClientInfo(credentials, token);
        final PlatformInfo updatedPI = this.platformData.getPlatforms().get(token);
        return updatedPI.getVersions() List<VersionDetails>
                .stream() Stream<VersionDetails>
                .filter(v -> v.getVersion().equals(updatedPI.getCurrentVersion()))
                .findFirst() Optional<VersionDetails>
                .flatMap(vd -> vd.getEndpoints().stream()
                        .filter(e -> e.getIdentifier().equals(ModuleID.credentials) && e.getRole().equals(otherPlatformRole))
                        .findFirst()
                );

    } else throw new PlatformAlreadyRegistered();
}
```

Fonte: Próprio Autor

É importante notar que as exceções e a chave de abertura da função foram emitidas por questões de legibilidade, as exceções são as mesmas do método "*registerAsSender*" ilustrado na Figura 26.

A função faz a checagem se a plataforma já está registrada no sistema, caso esteja, a exceção "*PlatformAlreadyRegistered*" é disparada. Caso contrário, um objeto "*PlatformInfo*" é criado e armazenado em um dicionário que possui como chave o token que esta plataforma utiliza para autenticação. Após isso a função "*retrieveClientInfo*" é chamada.

Figura 28 - Função "retrieveClientInfo"



Fonte: Próprio Autor

Esta função tem como objetivo atualizar os dados da plataforma. Através do objeto de credenciais, é feita uma requisição para o *endpoint* de versões, que retorna todas as versões suportadas pela plataforma, com essas versões em mãos, é selecionada a versão mútua mais atual entre as plataformas e é feita uma requisição para obter os detalhes desta versão específica. Após isso os atributos de versões e versão atual da plataforma são atualizados no dicionário.

Até aqui os primeiros dois passos gerais estabelecidos foram cumpridos. Continuando na função "*getCredentialsEndpoint*", o objeto "*PlatformInfo*" agora atualizado é obtido e o *endpoint* de credenciais da plataforma é retornado. Cada *endpoint* tem um papel, como, por exemplo, o *endpoint* de credenciais tem o papel de "*Sender*" ou "*Receiver*", e é através do papel passado por parâmetro na função que o *endpoint* desejado é retornado.

Voltando na função “*registerAsSender*” ilustrada na Figura 26, através da função “*getCredentialsEndpoint*” foi obtido o *endpoint* de credenciais da plataforma. Seguindo o processo na Figura 6, o próximo passo é gerar o “CREDENTIALS_TOKEN_B” e enviá-lo para o *endpoint* de credenciais. A variável “*platform*” representando a plataforma que enviou a requisição é inicializada, o token é armazenado em uma lista que contém apenas token válidos, ou seja, o token está sendo considerado válido.

Uma nova função é chamada, a “*senderLogic*”, que é o passo final da parte do enviador. A função exibida na Figura 29 completa os dois últimos passos gerais para a implementação do processo de registro por parte do enviador. Aqui os dados são definidos e enviados em uma requisição para o receptor, o retorno dessa requisição é a credencial que deverá ser utilizada para autenticações com esta plataforma.

Figura 29 - Função "senderLogic"

```
@SneakyThrows  1 usage  luizf +1 *
public Credentials senderLogic(final Endpoint endpoint, final String tokenB) {
    final var credentialsRole = CredentialsRole.builder()
            .role(Role.CPO)
            .partyId(new CiString( value: "PSI"))
            .countryCode(new CiString( value: "BR"))
            .build();
    final var credentials = Credentials.builder()
            .url(new URL( spec: this.platformData.getServerUrl() + "/ocpi/cpo/versions"))
            .token(this.credentialsTokenService.encodeToken(tokenB))
            .roles(Arrays.asList(credentialsRole))
            .build();
    final var platformInfo = this.platformData.getPlatforms().get(tokenB);
    final var objectMapper = new ObjectMapper();
    final var typeReference = new TypeReference<Response<Credentials>>(){};  new *
    final var otherPlatformCredentialsResponse = objectMapper.readValue(httpRequest(endpoint.getUrl(),  method: "POST", platformInfo.getToken(), credentials), typeReference);
    final var otherPlatformCredentials = otherPlatformCredentialsResponse.getData();
    final var tokenC = otherPlatformCredentials.getToken();
    platformInfo.setToken(this.credentialsTokenService.decodeToken(tokenC));
    return credentials;
}
```

Fonte: Próprio Autor

O token de “*PlatformInfo*” agora é atualizado e o objeto de credenciais é retornado.

Na função “*registerAsSender*” a credencial retornada é armazenada, dessa forma concluindo o último passo necessário para o enviador.

3.6.2 RECEIVER

De acordo com o OCPI, o processo de registro de credenciais por parte do receptor ou “*Receiver*” pode ser fragmentado em passos mais generalizados, que são:

1. Gerar um token e enviar ao enviador de forma offline
2. Armazenar o token recebido na requisição
3. Obter os dados do enviador e armazenar
4. Gerar um novo token e enviar ao validador

Da mesma forma que foi analisada a implementação do enviador agora será analisada a implementação do receptor, que é o outro lado do processo.

Na Figura 30, as exceções e a abertura de chave do método foram cortadas por motivos de legibilidade. Através do método “*getCredentialsEndpoint*”, o *endpoint* de credenciais da plataforma que enviou a requisição é obtido, após isso é feita a geração do “tokenC” seguido de sua validação. A plataforma que está armazenada no sistema é atualizada com este novo token e o primeiro token que foi criado é invalidado (A criação do primeiro token não consta nessa implementação pois ela ocorre de forma offline como determinado pelo OCPI).

Figura 30 - Método "registerAsReceiver"

```
public Credentials registerAsReceiver(final Credentials credentials, final String tokenB) throws PlatformAl
    final var credentialsEndpointOpt = getCredentialsEndpoint(credentials, tokenB, InterfaceRole.SENDER);
    var tokenC = "";
    final var platform = this.platformData.getPlatforms().get(tokenB);
    if(credentialsEndpointOpt.isPresent()) {
       tokenC = credentialsTokenService.generateToken();
       this.credentialsTokenService.validateToken(tokenC);
       this.platformData.getPlatforms().remove(tokenB);
       this.platformData.getPlatforms().put(tokenC, platform);
       this.credentialsTokenService.invalidateToken(this.tokenA);
    }
    final var credentialsRole = CredentialsRole.builder()
            .role(Role.CPO)
            .partyId(new CiString( value: "PSI"))
            .countryCode(new CiString( value: "BR"))
            .build();
    final var platformCredentials = Credentials.builder()
            .url(new URL( spec: this.platformData.getServerUrl() + "/ocpi/cpo/versions"))
            .token(this.credentialsTokenService.encodeToken(tokenC))
            .roles(Arrays.asList(credentialsRole))
            .build();
    platform.setCredentialsUsed(platformCredentials);
    platform.setCredentials(platformCredentials);
    System.out.println(platform);
    var encodedToken = this.credentialsTokenService.encodeToken(platform.getToken());
    System.out.println("Token codificado: " + encodedToken);
    return platformCredentials;
}
```

Fonte: Próprio Autor

A criação do objeto de credenciais é feita, armazenada no sistema e retornada do método. Dessa forma o enviador consegue armazenar os dados de credenciais do receptor e o receptor já possui os dados do enviador armazenados. Tanto o receptor como o enviador possuem armazenados no sistema tokens válidos para acessar os *endpoints* um do outro.

4 TESTES DO PROCESSO DE REGISTRO DO OCPI

Para a realização dos testes foi utilizado o *Postman*, que é um software que possibilita o envio de requisições HTTP para as aplicações.

Esse capítulo será dividido em duas partes, sendo cada uma um cenário diferente do processo de registro. As seguintes configurações serão válidas para todos os casos de teste:

- Tanto o receptor quanto o enviador rodam em "*localhost*"
- A porta da aplicação do receptor é 8080
- A porta da aplicação do enviador é 8092

Como o OCPI determina que o primeiro token seja passado de forma offline, nesta implementação há um *endpoint* que retorna um token válido, desta forma é possível ter acesso ao primeiro token. Por questões de simplificação e que fogem do escopo deste trabalho, para este *endpoint* não é necessário passar pelo filtro de autenticação.

Os tokens serão denominados como está descrito no processo de registro, determinado pelo OCPI, que está ilustrado na Figura 6.

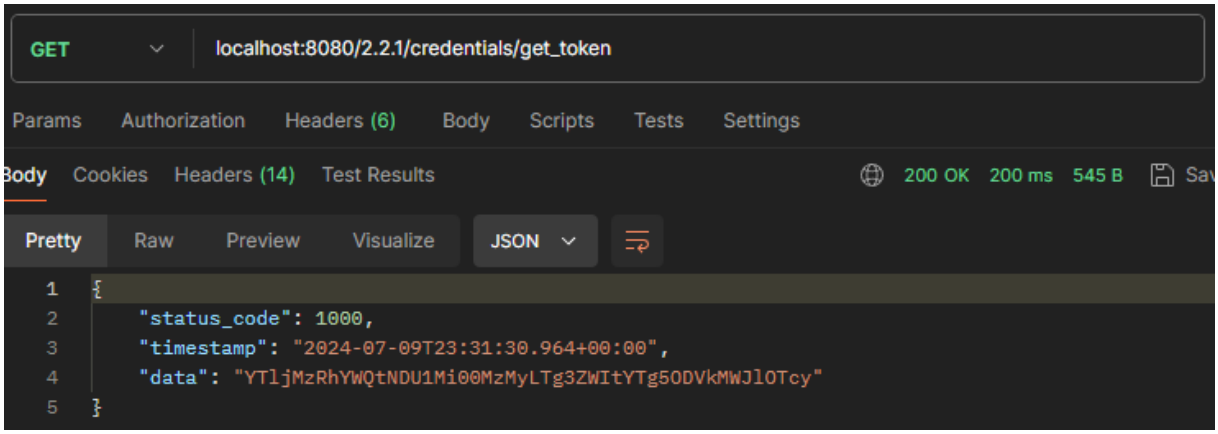
- CREDENTALS_TOKEN_A será Token A
- CREDENTIALS_TOKEN_B será Token B
- CREDENTIALS_TOKEN_C será Token C

4.1 TESTE DE CASO POSITIVO

Esse teste representa o processo de registro entre duas plataformas que devem ser registradas sem erros.

A Figura 31 exibe a resposta da requisição para a geração do Token A, que é o atributo "data". Esta é a forma que o receptor gera o Token A.

Figura 31 - Geração do Token A



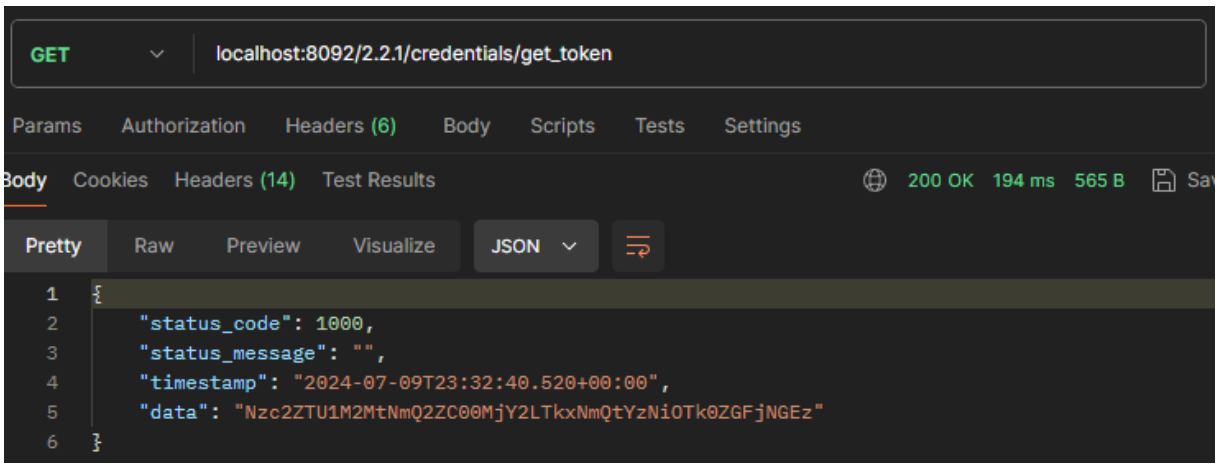
Fonte: Próprio Autor

O OCPI sugere o envio deste token para o enviador por uma forma segura. A estratégia utilizada para os testes foi de enviar o objeto de credenciais para o *endpoint* de credenciais do enviador, que dá início ao processo de registro. Para acessar esse *endpoint* é necessário possuir um token válido. Para conseguir o token válido a estratégia utilizada foi utilizar o *endpoint* que retorna um token válido, o mesmo utilizado para a geração do Token A, porém, dessa vez, para o enviador.

Essa parte da autenticação é livre para cada plataforma implementar, é interno de cada plataforma, o receptor não atua nessa ação. Como meio de simplificação, a estratégia da obtenção desse token foi através do mesmo método para retorno do Token A, tendo em vista que, esta questão não faz parte do OCPI, e está fora do escopo deste trabalho. Porém, em aplicações reais, é recomendável que a autenticação seja executada de forma segura.

A Figura 32 mostra a forma da geração do token para autenticação do enviador, é a partir deste token obtido através da requisição que a autenticação é feita.

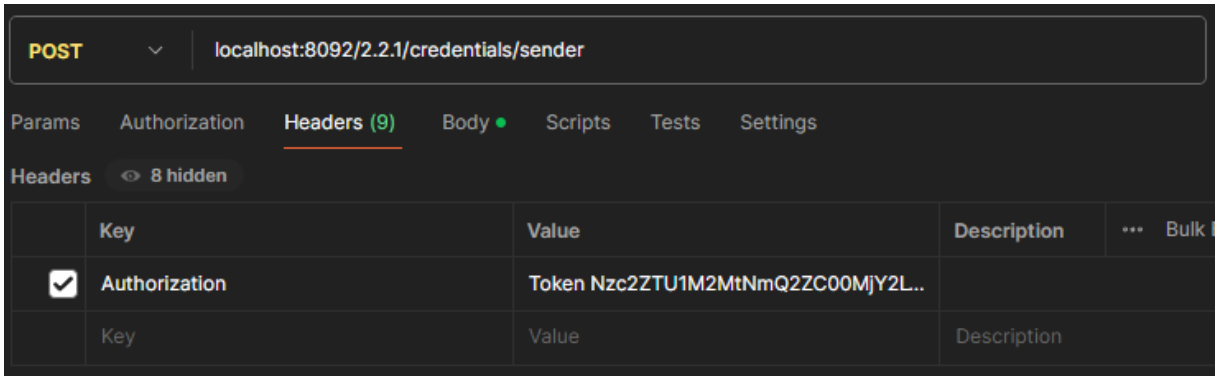
Figura 32 - Geração do Token Interno do Enviador



Fonte: Próprio Autor

Com este token agora é possível acessar colocando-o no cabeçalho da requisição para o módulo de credenciais do enviador como mostra a Figura 33.

Figura 33 - Cabeçalho da Requisição do Processo de Registro do Enviador

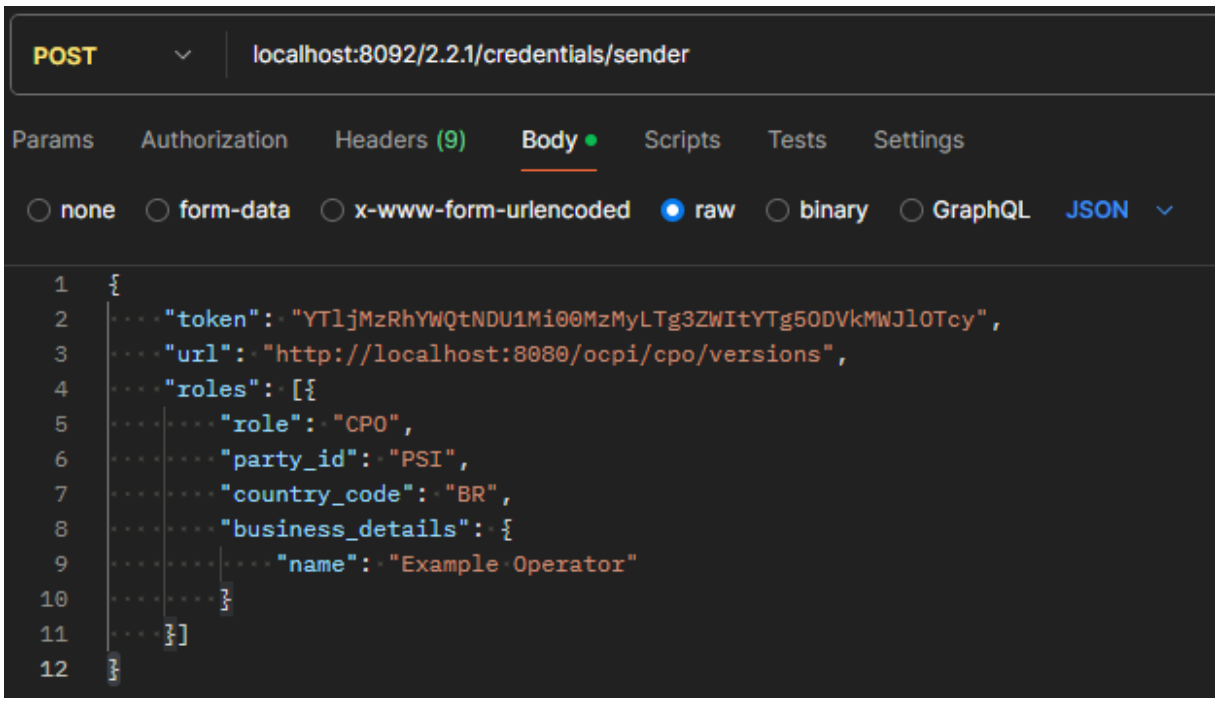


Fonte: Próprio Autor

O token é adicionado na chave “*Authorization*” e é precedido da palavra “Token”.

A Figura 34 mostra o objeto de credenciais do receptor no corpo da requisição para começar o processo de registro por parte do enviador.

Figura 34 - Corpo da Requisição do Processo de Registro do Enviador

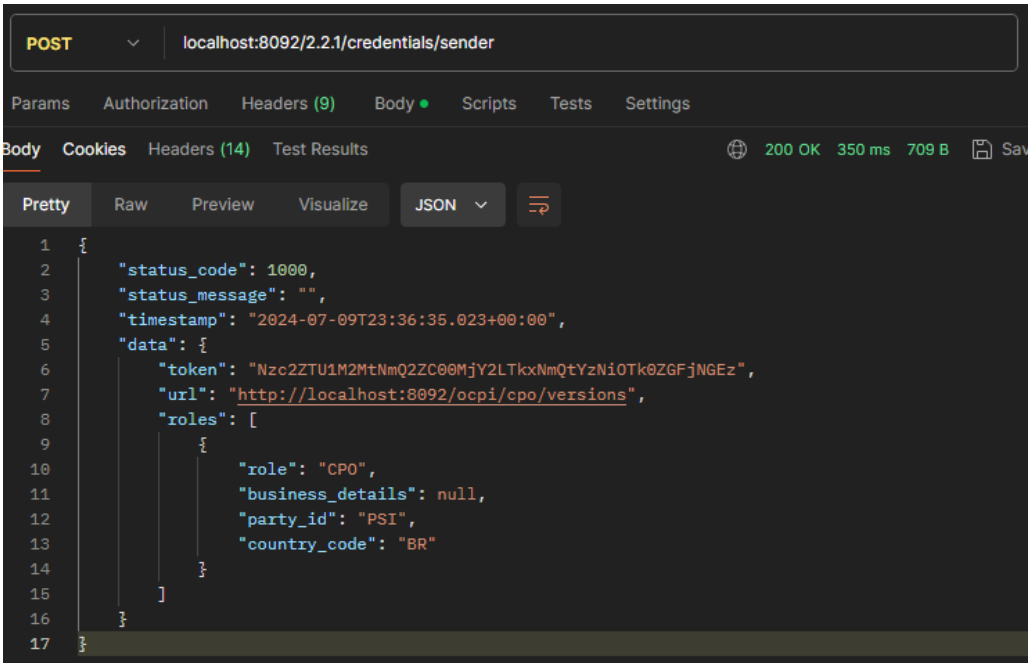


Fonte: Próprio Autor

O corpo da requisição consiste no objeto de credencial do receptor, que contém o Token A, esse é o objeto que deve ser passado de forma *offline*, e com ele o enviador começa o processo de registro.

A Figura 35 mostra a resposta do enviador ao registro, contendo o seu objeto de credenciais final.

Figura 35 - Resposta do Enviador a Requisição de Registro de Credenciais



Fonte: Próprio Autor

O status “1000” indica que o processo de registro ocorreu com sucesso.

Na Figura 36, que exibe os logs do enviador, é importante notar que a imagem está cortada em prol da legibilidade. A penúltima linha contém o objeto “*PlatformInfo*” com as informações da plataforma receptora, o token exibido está codificado, na linha abaixo está o token codificado, que é esse o utilizado para enviar no cabeçalho das requisições.

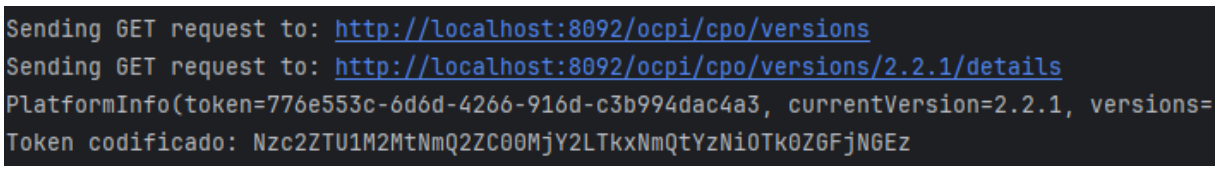
Figura 36 - Logs do Enviador

```
Sending GET request to: http://localhost:8080/ocpi/cpo/versions
Sending GET request to: http://localhost:8080/ocpi/cpo/versions/2.2.1/details
Sending POST request to: http://localhost:8080/2.2.1/credentials/receiver
PlatformInfo(token=ba217279-3eb0-4604-b415-96b6432f2d35, currentVersion=2.2.1, versions=[Ve
Token codificado: YmEyMTcyNzktM2ViMC00NjA0LWI0MTUtOTZiNjQzMmYyZDM1
```

Fonte: Próprio Autor

A Figura 37 exibe os logs do receptor.

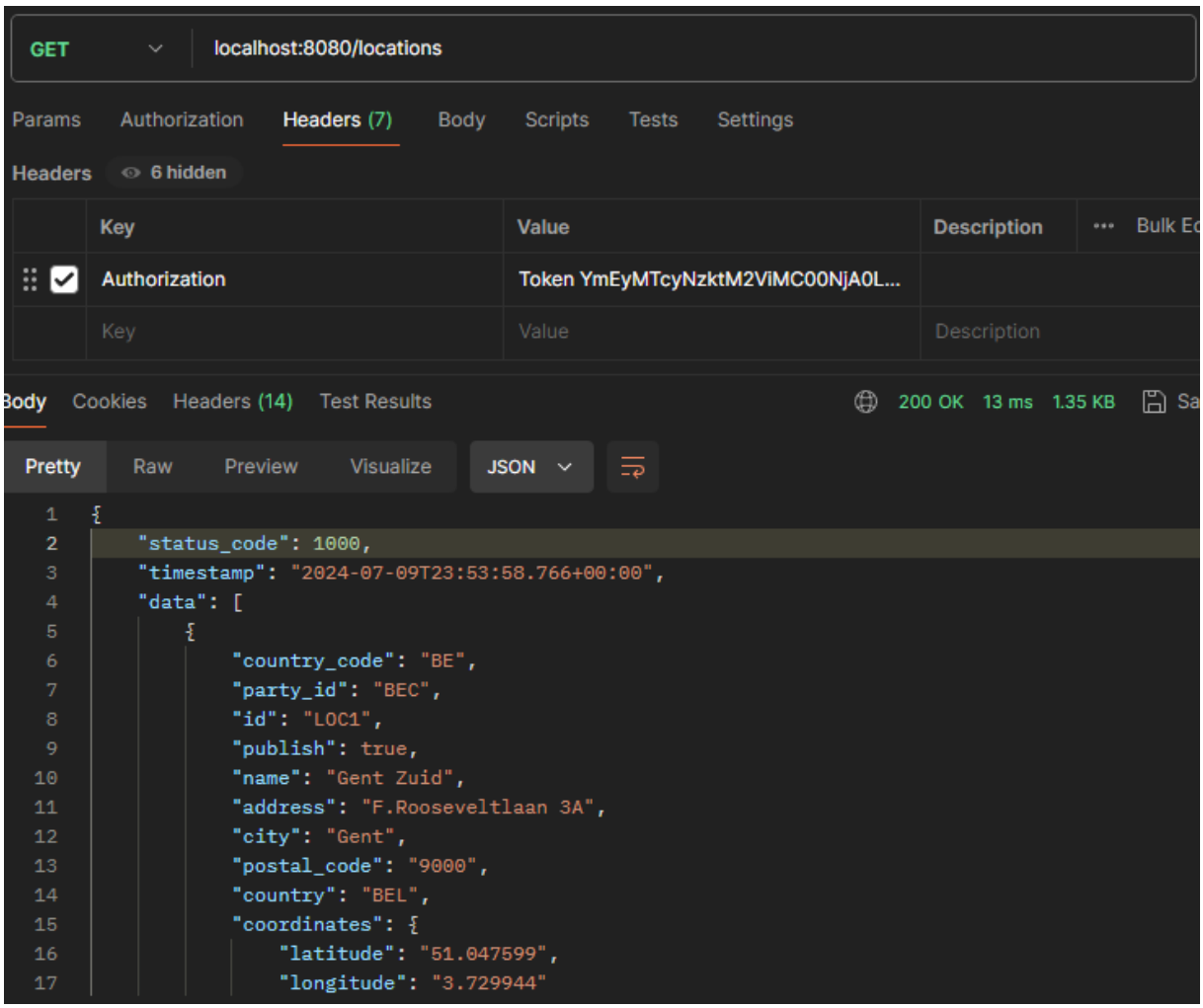
Figura 37 - Logs do Receptor



Fonte: Próprio Autor

Agora com os tokens codificados, é possível testar o acesso a algum *endpoint* da aplicação. A Figura 38 exibe a requisição ao *endpoint* do módulo de Localização da aplicação que roda na porta 8080.

Figura 38 - Resposta de Requisição para Módulo de Localização

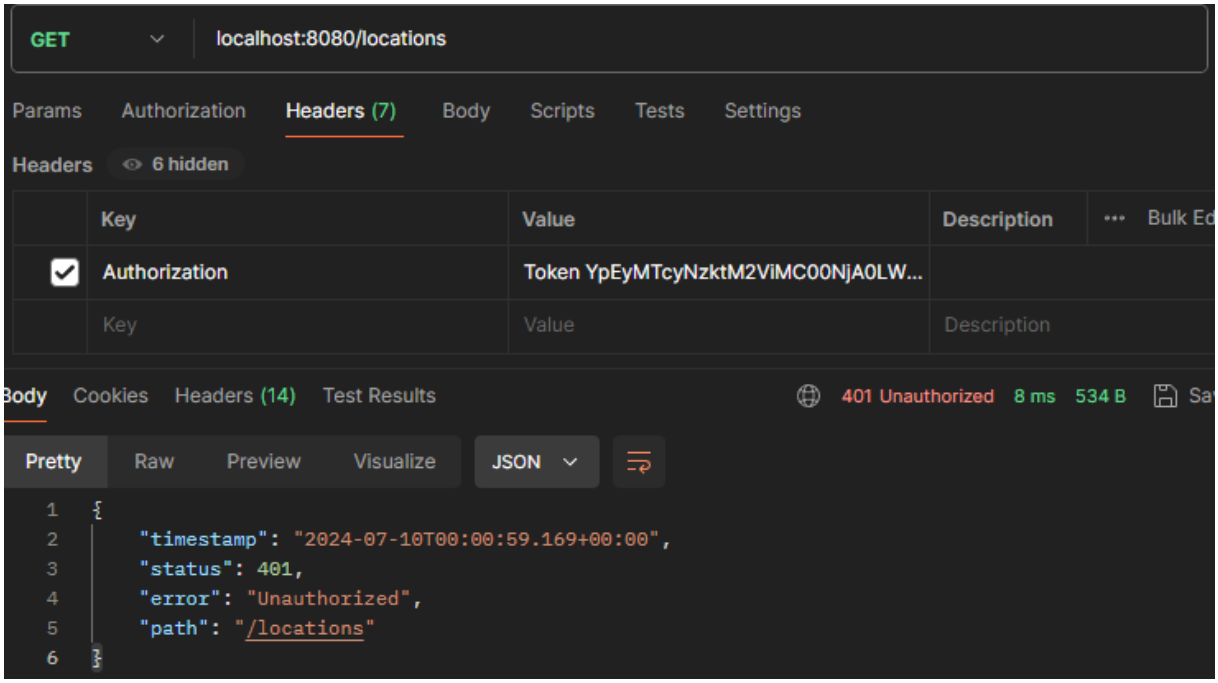


Fonte: Próprio Autor

O token utilizado no cabeçalho é o mesmo token que foi armazenado no sistema, porém codificado. Com o registro, foi possível acessar o *endpoint* de Localização da plataforma e obter a lista de localizações.

A Figura 39 ilustra o comportamento caso o token seja inválido.

Figura 39 - Requisição com Token Inválido



Fonte: Próprio Autor

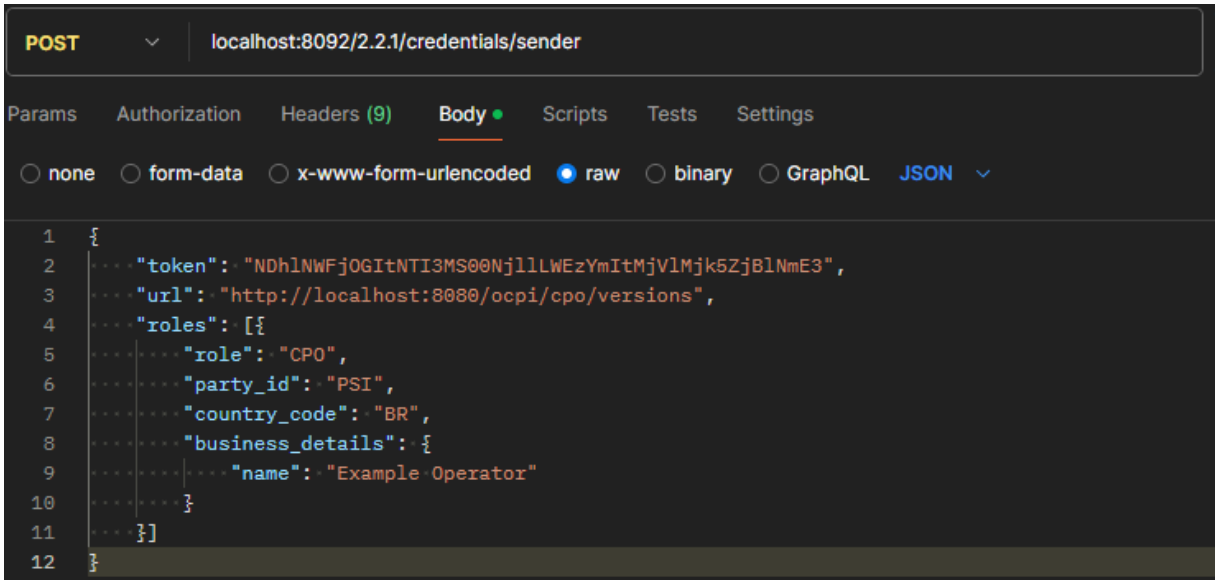
A segunda letra do token original foi alterada e resultou em uma resposta de erro na requisição, com o *status* 401, que significa não autorizado.

4.2. REGISTRO DE PLATAFORMA PREVIAMENTE REGISTRADA

Utilizando o mesmo exemplo do capítulo 4.1, será observado o comportamento da aplicação caso uma plataforma que já foi registrada, tentar fazer o processo de registro.

A Figura 40 mostra o objeto de credencial que será utilizado para o processo de registro.

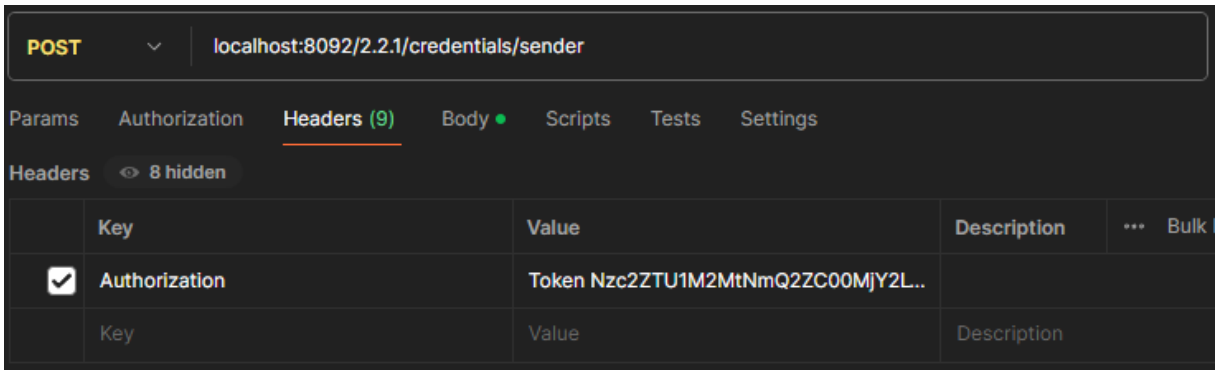
Figura 40 - Corpo da Requisição de Registro para uma Plataforma Previamente Registrada



Fonte: Próprio Autor

Esse objeto de credenciais que está no corpo da requisição já foi utilizado para o processo de registro, por isso o resultado esperado é que todo processo não ocorra novamente. A Figura 41 mostra o que está sendo enviado no cabeçalho da requisição

Figura 41 - Cabeçalho da Requisição de Registro de uma Plataforma Previamente Registrada

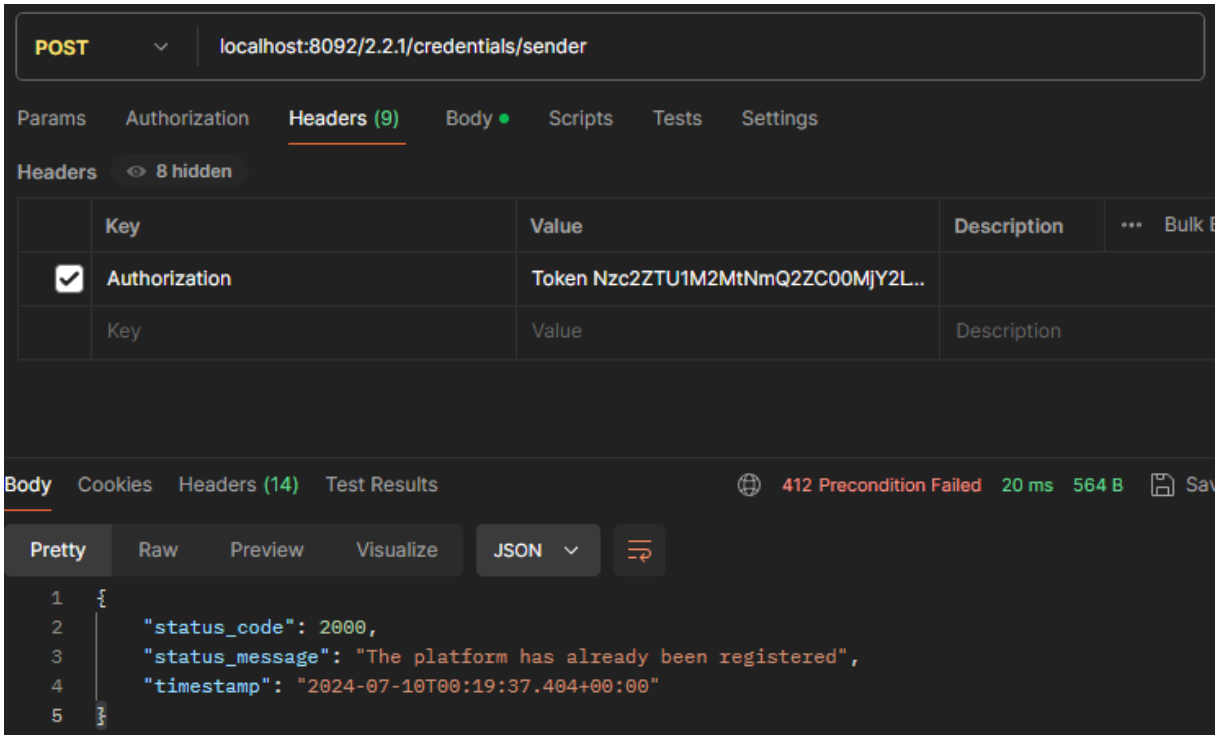


Fonte: Próprio Autor

Desta vez, o token utilizado no cabeçalho é o token para acessar a aplicação que roda na porta 8092, este é um token válido, pois é o token que foi gerado no processo de registro já executado.

A Figura 42 mostra o comportamento desta requisição, é retornado uma resposta OCPI com *status* 2000, que significa erro, com a mensagem indicando que a plataforma já foi registrada, sendo assim, impedindo que prossiga com o registro.

Figura 42 - Resposta da Requisição de Registro para Plataforma Previamente Registrada



Fonte: Próprio Autor

5 CONSIDERAÇÕES FINAIS

Ao longo deste trabalho, foi dedicada atenção ao aprofundamento teórico da infraestrutura de comunicação no contexto da mobilidade elétrica.

Foi explorado o conceito de roaming e sua relevância no cenário da mobilidade elétrica, bem como sua importância na garantia de uma experiência fluida e eficiente para os usuários. Além de destacar a necessidade de uma comunicação interoperável entre diferentes serviços de carregamento.

O protocolo OCPI foi abordado com foco, ilustrando como que, através dele, a interoperabilidade é atingida através de seus módulos e como funcionam as requisições entre CPOs e eMSPs com a utilização deste protocolo.

O processo de desenvolvimento do OCPI foi discutido e a implementação foi abordada com detalhamento. O processo de registro, demonstrou ser um componente crucial para garantir a segurança e confiabilidade na comunicação entre as diferentes partes envolvidas, ilustrando o seu papel no protocolo.

Em síntese, este estudo oferece uma compreensão da infraestrutura de comunicação na mobilidade elétrica, destacando desafios, soluções e a importância de protocolos como o OCPI, proporcionando uma base sólida para futuras implementações e expansões do que foi mostrado.

A expectativa é de que as informações apresentadas nesse trabalho contribuam para o avanço do conhecimento nesse campo e inspirem futuras pesquisas e inovações, tanto como, a adoção desse protocolo por serviços da mobilidade elétrica.

REFERÊNCIAS

ASSOCIAÇÃO BRASILEIRA DO VEÍCULO ELÉTRICO (ABVE). **Série Histórica**. Disponível em: <https://abve.org.br/serie-historica/ >. Acesso em: 14 ago. 2023.

INTERNATIONAL ENERGY AGENCY (IEA). **Global EV Outlook 2023: Prospects for Electric Vehicle Deployment**. Disponível em: < https://www.iea.org/reports/global-ev-outlook-2023/prospects-for-electric-vehicle-deployment>. Acesso em: 14 ago. 2023.

DRIIVZ. **Range Anxiety**. Disponível em < https://driivz.com/glossary/range-anxiety/>. Acesso em: 16 ago. 2023.

DRIIVZ. **Charge Point Operator (CPO)**. Disponível em: < https://driivz.com/glossary/charge-point-operator/#:~:text=Charge%20Point%20Operator%20(CPO)%20is,optimal%20ongoing%20EV%20charging%20operations.>. Acesso em: 16 ago. 2023.

DRIIVZ. **Smart EV Charging and Energy Management: The Essential Guide.** Disponível em: <https://driivz.com/blog/ev-smart-charging-benefits/>. Acesso em: 10 out. 2023.

VAN DER KAM, Marti; BEKKERS, Rudi; **Achieving interoperability for EV roaming: Pathways to harmonization** p. 1-34, mai 2020.

VAN DER KAM, Marti; BEKKERS, Rudi; **Comparative analysis of standardized protocols for EV roaming** p. 1-41, mai 2020.

Electric Vehicles' Range Jumps to Top of Priorities for Consumers. Disponível em: <https://www.autolist.com/news-and-analysis/2021-survey-electric-vehicles>. Acesso em 15 out. 2023.

EVRoaming Foundation; **OCPI 2.2.** Disponível em: <https://evroaming.org/app/uploads/2021/11/OCPI-2.2.1.pdf>. Acesso em 02 nov. 2023

IEA. Topp 5 barriers to EV adoption reported by EV100 member companies. Paris: IEA, 2021. Disponível em: <https://www.iea.org/data-and-statistics/charts/top-5-barriers-to-ev-adoption-reported-by-ev100-member-companies>. Acesso em 13 ago. 2024.

MATOS, J.H *et al.* **Vehicle Recharging and Roaming Protocols: A Brazilian Electromobility Case Study**, Smart Innovation, Systems and Technologies, v.295, p. 334-342, jul 2022

TOLKIEN, J.R.R. **O Senhor dos Anéis: A Sociedade do Ane**l. 2. ed. São Paulo: Martins Fontes, 1954.

MATOS, J. H. J. G.; TANIGUCHI, F. K.; PARREIRA, A. D.; MARQUES, M. d. C.; JUNIOR, E. L. Electric Vehicle Recharging and Roaming Protocols: A Brazilian Electromobility Case Study. In: IANO, Y.; SAOTOME, O.; KEMPER VÁSQUEZ, G. L.; COTRIM PEZZUTO, C.; ARTHUR, R.; GOMES DE OLIVEIRA, G. (eds). Proceedings of the 7th Brazilian Technology Symposium (BTSym'21). BTSym 2021. Smart Innovation, Systems and Technologies, v. 295. Springer, Cham, 2022. Disponível em: <https://doi.org/10.1007/978-3-031-08545-1_32>. Acesso em: 13 ago. 2024.

EVRoaming Foundation, **Downloads**. Disponível em: < https://evroaming.org/downloads/>. Acesso em 17 jul. 2024.

EVRoaming Foundation, **DirectPayment Solution – OCPI 2.2.1**. Disponível em: < https://evroaming.org/app/uploads/2024/03/DirectPayment_2_2_1___EVRF_version.pdf>. Acesso em 17 jul. 2024.

EVRoaming Foundation, **Freshmile x Chargemap**. Disponível em: < https://evroaming.org/testimonials/freshmile-x-chargemap/>. Acesso em 17 jul. 2024.

GUILLEMIN, S. *et al*. **Electrical Vehicle Smart Charging Using the Open Charge Point Interface (OCPI) Protocol**, jun 2024.

SAFAYA, R., **The Limits of OCPI for Smart Charging**. Disponível em: < https://www.edrv.io/blog/the-limits-of-ocpi-for-smart-charging/>. Acesso em 18 jul. 2024.

UNIÃO EUROPEIA. Site oficial da União Europeia. Disponível em: < https://transport.ec.europa.eu/transport-themes/clean-transport/alternative-fuels-sustainable-mobility-europe/alternative-fuels-infrastructure_en>. Acesso em 18 jul 2024.

ENERGY-RETAIL-SOLUTIONS, SCHEIDT & BACHMANN PARTNERS WITH CHARGEPOINT. Disponível em: <https://www.scheidt-bachmann.de/en/article/scheidt-bachmann-partners-with-chargepoint>. Acesso em 18 jul 2024.