Sistema de Recarga de VEs Baseado em Blockchain

Caick Wendell Lopes dos Santos[1], **João Lucas Silva Serafim Silva**[1]

[1]Departamento de Tecnologia – Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n, Novo Horizonte / Feira de Santana – BA, Brasil- 44036-9

Resumo. A demanda dentro do contexto de inovação está atrelada a necessidade de priorizar alguns aspectos do produto. Neste sentido, as crescentes atualizações no contexto dos veículos elétricos põem em dúvida a confiabilidade e auditabilidade do cenário. O documento a seguir descreve o processo para implementação de tecnologia de *blockchain* em um sistema de recarga de VEs. O sistema utiliza de *blockchain* de tipo Ethereum para comunicações periódicas com o servidor. A aplicação permite o registro e sincronização de compras através de um banco de dados descentralizado.

Abstract. Demand within the context of innovation is linked to the need to prioritize certain aspects of the product. In this sense, the increasing updates in the context of electric vehicles raise questions about the reliability and auditability of the scenario. The following document describes the process for implementing blockchain technology in an EV charging system. The system uses Ethereum-type blockchain for periodic communications with the servers. The application enables the registration and synchronization of purchases through a decentralized database.

1. Introdução

O crescimento da indústria de veículos elétricos requer que cada vez mais demandas sejam atendidas de forma a tornar a experiência dos usuários e das empresas a mais confortável e efetiva possível. Entretanto, além das questões mais notáveis e tangíveis como o gerenciamento de rotas, a integração entre os sistemas e a escassez de auditabilidade acarretam na carência de confiabilidade.

Este relatório apresenta a descrição do processo de aprimoramento de um sistema de recarga de VEs. De forma que o mesmo adquira maior confiabilidade e auditabilidade através da implementação de tecnologia de *blockchain*. A aplicação permanece utilizando a linguagem Python em sua maior parte, com o auxílio dos protocolos *Message Queue Telemetry Transport*(MQTT) com broker Mosquitto e arquitetura *Rest*. Esta atualização conta com o uso da biblioteca W3 para Python e linguagem de programação Solidity. A aplicação foi projetada em sistemas Linux com o auxílio da IDE Visual Studio Code, simulador Ethereum Ganache e plataforma GitHub.

O programa conta com todas as funcionalidades de sua versão anterior e com a adição das funcionalidades de registro de compras em um banco de dados descentralizado e sincronização do mesmo entre os servidores. Este relatório está organizado da seguinte forma. A seção 2 contém o conhecimento teórico utilizado para o desenvolvimento da aplicação. A seção 3 apresenta as ferramentas e métodos utilizados. A seção 4 demonstra os resultados e falhas obtidos. Por último a seção 5 refere-se as conclusões obtidas ao término do projeto.

2. Fundamentação Teórica

A seção a seguir tem por objetivo fazer um breve detalhamento do conhecimento teórico necessário para obtenção do produto final.

2.1. *Blockchain*

A primeira execução do conceito de *blockchain* em uma a rede a vincula como base para o surgimento das criptomoedas e está presente no paper "Bitcoin: A Peer-to-Peer Electronic Cash System" por Satoshi Nakamoto[Di Pierro 2017]. A *blockchain* é uma tecnologia de registro distribuído que organiza dados em forma de blocos protegidos pro criptografia e interligados de forma a criar uma cadeia segura e imutável.

Funciona como um livro-razão composto por uma rede *peer-to-peer* e um banco de dados distribuído descentralizado.

Blockchains fazem uso de criptografia assimétrica (usualmente RSA) para verificar a autenticidade de requisições de transação em uma determinada porcentagem dos clientes conectados (consenso).

A nível individual de cada cliente da *blockchain*, o processo de verificação dar-se-a por meio de cifra de um artefato de verificação (geralmente um *hash* da requisição) com a chave denominada "privada"(assim chamada por não ser possível que seja facilmente derivada de seu par chamado "público"), o qual será decifrado, pelo cliente da *blockchain*, com uma "chave pública"associada a uma "chave privada"e comparada com a informação que foi utilizada para gerar a mensagem criptografada de verificação. Tal processo é denominado "assinatura digital".

No retorno da consulta de uma transação, a criptografia assimétrica é utilizada em seu modo "normal", isto é, com o uso da "chave pública"para realização da cifra da mensagem de resposta, enquanto que o cliente utilizará sua "chave privada"para decifrar a mensagem.

2.1.1. Contratos inteligentes

Um contrato inteligente é um código que roda em uma *blockchain* com o objetivo de facilitar, executar e impor os termos de um acordo entre partes não confiáveis[Alharby and Van Moorsel 2017].

2.1.2. Criptografia SHA-256

SHA-256 é um algoritmo de criptografia não-reversível (portanto seguro) que permite a transformação de uma entrada de dados quaisquer de tamanho até $(2^{64} - 1)$ bits em uma saída de 256 bits fixa. Tal saída é denominada "hash"[National Institute of Standards and Technology 2002].

3. Metodologia

A princípio, fizeram-se necessárias pesquisas que esclarecessem o comportamento e implementação da tecnologia de *blockchain* junto a discussões com o intuito de direcionar os primeiros passos com base na tentativa de prever possíveis alterações.

As discussões resultaram na conclusão de que a estrutura do projeto anterior seria mantida e o planejamento seguinte centralizou-se na adição de novas funcionalidades e integração das mesmas.

Ao analisar o funcionamento do estágio anterior da aplicação. Nota-se que grande parte das funcionalidades incluem o emprego de listas encadeadas, o planejamento inicial do projeto voltou-se para esta característica e então foram criados protótipos que desempenhassem funções semelhantes até que houvesse uma execução desejada.

Os protótipos giravam em torno do processo de envio e leitura de mensagens e impressão de strings em um terminal e serviram de ponte para compreender por completo o manuseio de *blockchains*. Os mesmos já contavam com contratos inteligentes escritos em linguagem Solidity compilados, implantados e executados em linguagem Python e testados através de simulador Ethereum Ganache.

O protótipo final detinha um contrato o qual continha uma lista de *strings* e era capaz de adicionar, ler e obter o tamanho da mesma.

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ListaCodificada {
    string[] private listaJson;

    event ListaAdicionada(string jsonData);

    function adicionarLista(string memory jsonData) public {
        listaJson.push(jsonData);
        emit ListaAdicionada(jsonData);
    }

    function obterLista(uint index) public view returns (string memory) {
        require(index < listaJson.length,'indice invalido');
        return listaJson[index];
    }

    function totalListas() public view returns (uint) {
        return listaJson.length;
    }
}
```

Figura 1. Contrato utilizado nas operações do sistema

O formato do contrato em seu protótipo final mostrou-se ideal para integração com o sistema e seu processo de incorporação foi iniciado.

O protótipo foi dividido em duas partes para melhor coesão. A implantação e as transações foram integradas separadamente, logo, faz-se necessário que o processo de implantação do contrato seja iniciado antes de qualquer servidor.

No que tange as funcionalidades novas, foram adicionadas duas funções relacionadas ao processo de registro de compras aos servidores: *addToBlockchain()* e *syncWithBlockchain()*.

A primeira é responsável por registrar na blockchain que uma compra foi feita e é executada em paralelo à adição da compra na base de dados local, enquanto que a segunda é executada a cada passagem, desde a inicialização do servidor, de um período pré-determinado (por padrão dez minutos) e é responsável por sincronizar o servidor com a *blockchain*, ou seja, atualizar os dados locais com base nos dados remotos compartilhados entre os clientes do *ledger* distribuído, permitindo a confiabilidade de dados entre servidores.

Uma mudança realizada sobre o sistema existente foi a segurança no armazenamento de credenciais de login. Para o sistema de recarga, tais credenciais de login são os IDs dos clientes (estações e veículos).

O algoritmo escolhido para fazer o armazenamento seguro das credenciais de login foi o *Secure Hash Algorithm 256* (SHA256), uma vez que esse algoritmo não só é complexo o bastante para ser considerado difícil de reverter seu funcionamento, mas é capaz de produzir saídas idênticas para entradas distintas, tornando impossível determinar, ao menos de forma automatizada, qual entrada gerou um *hash* de saída específico, mesmo que seja possível obter todos as entradas para uma saída.

No servidor, o armazenamento de informações é feito sempre fazendo referência ao *hash* SHA-256 de um ID, e também é essa informação passada criptografada à frente, quando requisitado.

No momento imediatamente antes da execução de uma requisição, no qual é verificada a autenticidade de uma credencial, a credencial do agente requisitante recebida pela servidor passa pelo processo de *hashing*, enquanto que credenciais de logins referenciadas são esperadas já estarem em formato de *hash*.

4. Resultados

Esta seção é dedicada a discutir os resultados obtidos durante a produção.

A atualização do produto demonstrou-se um sucesso e o sistema de recarga apresenta uma confiabilidade muito maior que a versão anterior. Os testes feitos englobavam os protótipos citados na seção 3 e a execução do sistema como um todo de maneira sequencial junto a averiguação do processo através das ferramentas disponíveis no simulador Ganache.

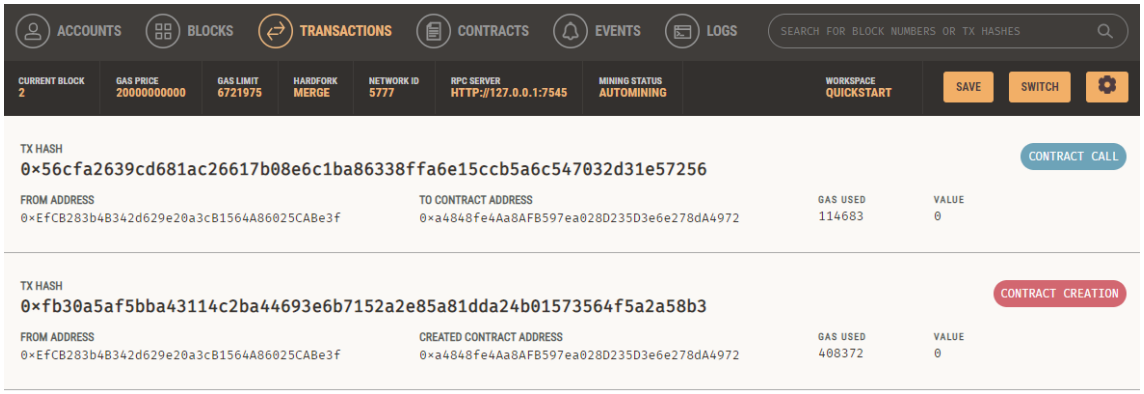
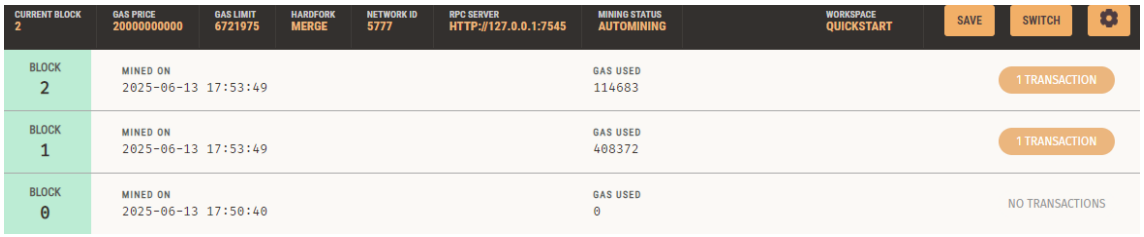


Figura 2. Aba de transações na interface do Ganache



(a) Aba de blocos na interface do Ganache

Portanto, é concluído que o sistema inclui as capacidades de registrar e sincronizar compras de forma descentralizada e confiável.

No quesito segurança de credenciais, percebe-se que o sistema também implementou de forma bem-sucedida a criptografia SHA256, fato esse constado ao comparar o ID real de cliente ao nome de seu arquivo na base de dados de um servidor.

Figura 3. ID como visto pelo cliente que é dono da credencial e enviado para o servidor como forma de autenticação

(a) Mesmo ID como é armazenado pelo servidor e repassado a outros clientes

5. Conclusão

O sistema é capaz de oferecer uma experiência muito mais segura ao usuário ao proporcionar a capacidade de auditoria confiável por conta da tecnologia de *blockchain* adicionada ao sistema de gerenciamento de compras.

O desenvolvimento da aplicação mostrou-se essencial para o aprendizado no contexto de manuseio de *blockchain*, além do exercício dos conceitos de confiabilidade, integridade e auditoria.

Um ponto de melhoria para possíveis atualizações seria a adição de recursos para registros de adição de reserva de maneira semelhante ao sistema de compra já existente.

Referências

Alharby, M. and Van Moorsel, A. (2017). Blockchain-based smart contracts: A systematic mapping study. *arXiv preprint arXiv:1710.06372*.

Di Pierro, M. (2017). What is the blockchain? *Computing in Science & Engineering*, 19(5):92–95.

National Institute of Standards and Technology (2002). Secure hash standard. *csrc.nist.gov/files/pubs/fips/180-2/final/docs/fips180-2.pdf*.