LemonBlock: Um Passo em Direção ao Futuro das Transações na UFPEL

Gustavo P. Pereira[1]**,Rávilon A. dos Santos**[2]

[1]Computação – Universidade Federal de Pelotas (UFPEL)
Pelotas – RS – Brazil

Resumo. *Este relatório apresenta o projeto LemonBlock, uma prova de conceito (PoC) que visa demonstrar a viabilidade da construção de uma blockchain personalizada para a Universidade Federal de Pelotas (UFPEL). LemonBlock foi concebida com o propósito de simular transações acadêmicas e administrativas dentro da universidade, com o objetivo de estabelecer um alicerce sólido para futuros desenvolvimentos. Este estudo demonstra não apenas a capacidade de criar uma infraestrutura de blockchain adaptada às necessidades da UFPEL, mas também explora a possibilidade de melhorar a eficiência, a transparência e a segurança das transações universitárias.*

1. Introdução

A Universidade Federal de Pelotas (UFPEL) é uma instituição de ensino superior de renome, comprometida em fornecer uma educação de qualidade e aberta à inovação. Em consonância com esse compromisso, apresentamos o projeto LemonBlock, uma prova de conceito (PoC) que visa demonstrar a viabilidade da construção de uma blockchain personalizada para a UFPEL.

Embora a UFPEL seja uma universidade pública, caracterizada por sua gratuidade, algumas transações financeiras e administrativas são inerentes à vida acadêmica. Isso inclui o pagamento de bolsas de estudo, bem como a alocação e gestão de recursos para viagens e programas de extensão, que são cruciais para enriquecer a experiência acadêmica dos estudantes. Além disso, a universidade oferece serviços de refeição em seus restaurantes universitários, onde os pagamentos precisam ser efetuados de maneira eficiente.

Neste relatório, descreveremos em detalhes o projeto LemonBlock, desde sua concepção até a implementação da prova de conceito. Vamos explorar como essa infraestrutura blockchain pode aprimorar a eficiência, a transparência e a segurança dessas transações acadêmicas, incluindo o gerenciamento de bolsas, a alocação de recursos para viagens ou programas de extensão e, também, a simplificação dos pagamentos feitos pelos alunos nos restaurantes universitários. Através deste estudo, esperamos não apenas mostrar a capacidade de criar uma solução adaptada às necessidades específicas da UFPEL, mas também abrir caminho para um desenvolvimento mais amplo e sério de uma blockchain que pode potencialmente transformar a maneira como a universidade gerencia suas operações e se relaciona com sua comunidade acadêmica, contribuindo para uma administração mais eficaz e transparente.

2. Benefícios do Uso da Blockchain para Transações na UFPEL

A implementação da blockchain, como parte do projeto LemonBlock, oferece uma série de benefícios significativos para a gestão de transações na Universidade Federal de Pelotas

(UFPEL). Abaixo, destacamos alguns dos principais benefícios:

1. **Transparência e Auditoria:** A blockchain é um registro digital imutável e transparente de todas as transações. Isso garante que todas as partes interessadas, incluindo estudantes, administradores e órgãos reguladores, tenham acesso a informações precisas e atualizadas. Qualquer transação pode ser auditada de forma transparente, o que reduz o risco de erros ou fraudes.
2. **Redução de Intermediários:** A blockchain permite a realização de transações peer-to-peer (P2P), eliminando a necessidade de intermediários, como bancos ou processadores de pagamento. Isso reduz custos e atrasos associados a terceiros.
3. **Eficiência e Automatização:** Contratos inteligentes podem ser implementados na blockchain para automatizar muitos aspectos das transações, como a concessão de bolsas de estudo, distribuição de recursos para viagens ou extensão, e até mesmo o processamento de pagamentos nos restaurantes universitários. Isso economiza tempo e recursos administrativos.
4. **Segurança de Dados:** A blockchain utiliza criptografia avançada e consenso descentralizado para garantir a segurança das informações. Isso protege os dados dos estudantes e da instituição contra ameaças de segurança, como violações de dados.
5. **Histórico Completo e Rastreabilidade:** Cada transação é registrada de forma permanente na blockchain, criando um histórico completo e rastreável. Isso pode ser valioso para fins de prestação de contas e para investigações de auditoria.
6. **Redução da Burocracia:** A blockchain simplifica muitos processos burocráticos, como o preenchimento de formulários e a obtenção de aprovações, através de contratos inteligentes e fluxos de trabalho automatizados.

Ao aproveitar os benefícios da tecnologia blockchain, o projeto LemonBlock da UFPEL tem o potencial de melhorar significativamente a gestão de transações acadêmicas e administrativas, promovendo eficiência, transparência e segurança em toda a universidade. Além disso, abre as portas para futuros desenvolvimentos e inovações na forma como a UFPEL interage com sua comunidade acadêmica e administra suas operações.

3. Casos de Uso

A construção dos casos de uso visava uma estrutura mais próxima do que seria desejado no sistema real. Englobando mecanismos de queima de tokens (burn) e geração (mint) onde a instituição geraria os tokens de acordo a quantidade de fundos destinadas para um determinado período, semestre por exemplo, assim dispensando a necessidade de transferir os fundos diretamente por terceiros e desburocratizando essa relação instituição - aluno. Quanto ao uso no dia-a-dia por parte dos estudantes eles poderiam depositar fundos na universidade para receber mais moedas se necessário para planejar seus pagamentos no RU. Os parceiros da universidade também deveriam ter suas próprias carteiras e na operação de saque dessas moedas para moeda corrente, podendo ser realizada apenas por essas parceiras por meio da universidade e nunca diretamente, seus tokens seriam ”queimados” para manter os fundos de moeda corrente da universidade coerentes com os tokens em circulação.

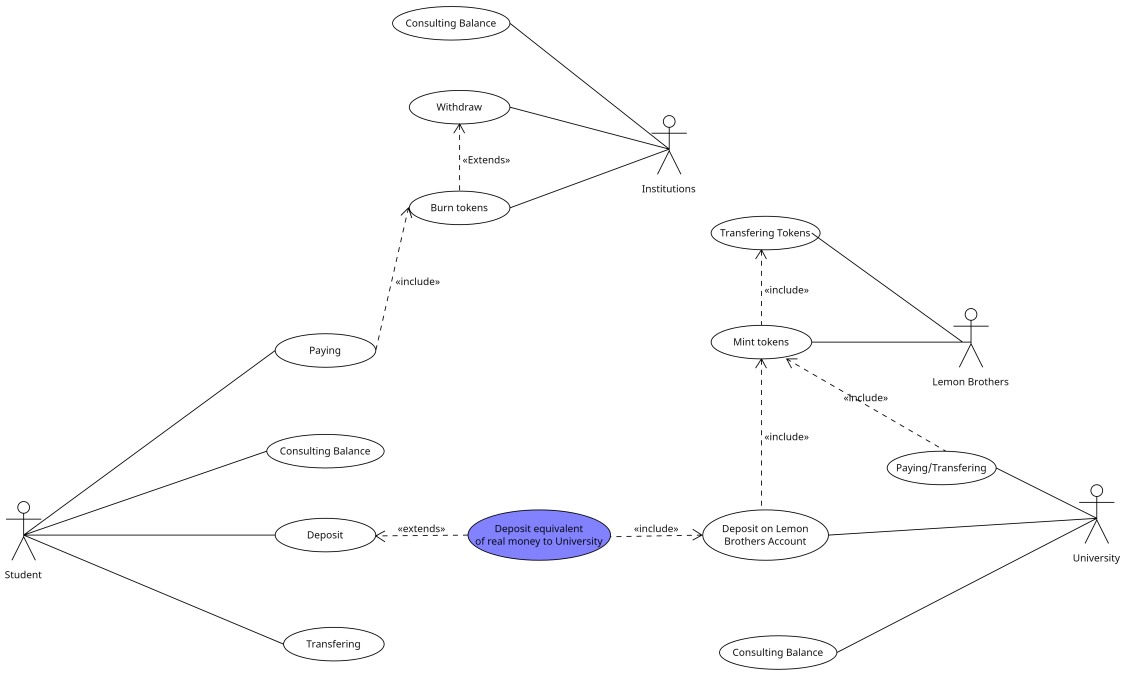

Figure 1. Casos de Uso

3.1. Obejtivos Gerais da POC

Utilizamos como referência a série de artigos [Chausier 2022] onde ele trata de forma didática sobre a criação de uma blockchain com Python seguindo como referência o Bit-Coin Considerando o escopo do projeto e o tempo hábil para produção do mesmo optamos por reduzi-lo a um MVP (Mínimo Produto Viável) considerando alguns subprodutos como essenciais, sendo eles: a Estrutura básica da blockchain e uma interface para uso.

4. Estrutura Básica da Blockchain

4.1. Blockchain

1. **Hash Criptográfico do Bloco Anterior:** Em uma blockchain, cada bloco armazena um hash criptográfico (uma sequência única de caracteres) do bloco anterior. Isso cria uma conexão encadeada entre os blocos, garantindo a integridade da cadeia. Se alguém tentar alterar os dados em um bloco anterior, isso alteraria o seu hash, tornando a adulteração evidente em toda a cadeia.
2. **Registro de Data e Hora:** Cada bloco contém um registro de data e hora que indica quando as transações ou dados foram adicionados ao bloco. Isso fornece um histórico imutável e cronológico das atividades na blockchain, o que é essencial para rastrear eventos e garantir a ordem das transações.
3. **Dados de Transações:** Os blocos também armazenam dados de transações. Em uma blockchain de criptomoeda, essas transações geralmente representam transferências de moedas entre contas. No entanto, em blockchains empresariais ou outras aplicações, os dados podem variar amplamente, incluindo contratos inteligentes, registros de propriedade, ou qualquer outra informação que precise ser registrada e compartilhada de forma segura.

4.2. Merkle Trees (Árvores de Merkle)

São estruturas de dados na forma de árvores binárias nas quais cada nó pai é o hash criptográfico dos nós filhos. Essas árvores são usadas para agrupar transações ou dados de maneira eficiente e criar um resumo (hash) que representa todas as informações contidas nas folhas da árvore.

Uso de Merkle Trees em Blockchains

1. **Economia de Espaço em Disco:** À medida que novas transações são adicionadas a um bloco em uma blockchain, é importante garantir que o tamanho dos blocos não cresça indefinidamente, pois isso aumentaria o requisito de armazenamento em disco.
2. **Eliminação de Transações Antigas:** Conforme novas transações são confirmadas e mais blocos são adicionados à blockchain, as transações antigas que já foram confirmadas por um número suficiente de blocos podem ser consideradas seguras e não precisam mais ser armazenadas em sua forma original.
3. **Uso de Merkle Trees em Compactação:** Para permitir a eliminação de transações antigas sem comprometer a integridade da blockchain, as transações em um bloco são organizadas em uma Merkle Tree. O hash da raiz dessa árvore (o Merkle Root) é incluído no cabeçalho do bloco, tornando-se parte integrante do cálculo do hash do bloco. Como resultado, qualquer alteração nas transações individuais na árvore afetaria o Merkle Root e, consequentemente, o hash do bloco.
4. **Compactação de Blocos Antigos:** Quando transações antigas podem ser eliminadas com segurança, os blocos antigos podem ser compactados, removendo ramos (ou seja, transações) da Merkle Tree. Os hashes intermediários (interior hashes) na árvore não precisam ser armazenados, economizando espaço em disco.

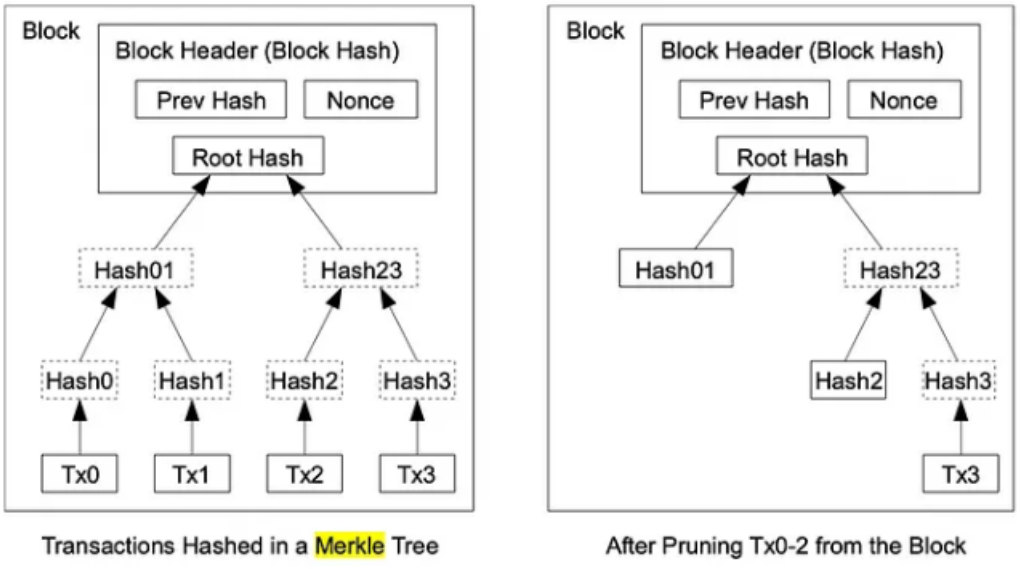

Figure 2. Merkle Tree

4.3. Transações e Segurança

Uma transação é composta por inputs e outputs, sendo os inputs compostos por uma chave pública do usuário que está fazendo a transação e o index da transação em questão, enquanto no output temos o address da carteira para onde vamos enviar. Isso segue o modelo de UTXO (Unspent Transaction Output) que é ultilizado por blockchains como Bitcoin cujo funcionamento é descrito por [Nakamoto 2008].

4.3.1. UTXO - Unspent Transaction Output

UTXO significa Unspent Transaction Output e é um conceito fundamental na tecnologia blockchain, especialmente em criptomoedas como o Bitcoin. UTXOs são usados para acompanhar a propriedade e a disponibilidade de unidades de criptomoeda (por exemplo, bitcoins) na blockchain. Para entender UTXOs, vamos dividir o termo:

Unspent (Não Gasto): Isso se refere a unidades de criptomoeda que ainda não foram usadas em uma transação. Em outras palavras, são fundos que ainda não foram enviados para outro usuário ou endereço. Quando você recebe criptomoeda, ela se torna uma saída não gasta até que você decida usá-la em uma transação.

Transaction (Transação): No contexto de uma blockchain, uma transação é uma operação que transfere criptomoeda de uma parte para outra. É um registro de transferência de valor, muitas vezes contendo entradas (referências a UTXOs previamente recebidos) e saídas (novos UTXOs criados).

Output (Saída): Uma saída é parte de uma transação que designa para onde a criptomoeda será enviada. Cada saída contém uma quantidade de criptomoeda e um endereço para onde ela deve ser enviada. Quando uma transação é confirmada, suas saídas se tornam novos UTXOs.

Aqui está como os UTXOs funcionam:

- Quando alguém lhe envia criptomoeda, isso cria um novo UTXO em seu nome. Este UTXO representa a quantidade de criptomoeda que você recebeu e está associado ao seu endereço.
- Quando você deseja gastar sua criptomoeda, cria uma nova transação. Esta transação faz referência a um ou mais de seus UTXOs (entradas) e especifica para onde a criptomoeda deve ser enviada (saídas). A soma das entradas deve ser igual ou maior do que a soma das saídas para manter o equilíbrio.
- Após a confirmação da transação e sua inclusão na blockchain, as entradas (UTXOs) que você usou são marcadas como ”gastas” e novos UTXOs são criados para os destinatários da criptomoeda.
- O modelo UTXO ajuda a manter a transparência e a segurança da blockchain. Qualquer pessoa pode verificar o histórico de um UTXO para garantir que ele não tenha sido gasto mais de uma vez. Isso é crucial para prevenir fraudes e manter a integridade do sistema de criptomoedas.
- UTXOs também contribuem para a privacidade, porque cada transação pode criar várias saídas, tornando mais difícil rastrear o fluxo de fundos.

Processo de Transação:

A transação é assinada durante a fase de input e temos a validação por meio do nó principal a partir da chave privada que permite o envio de ativos.

1. Assinatura (Signing): Essa etapa utiliza criptografia RSA como base tendo o dono da carteira as chaves pública e privada, antes de concluir uma transação deve se assinar a mesma.
2. Transmissão (Broadcasting):
    - Após assinar a transação, a carteira envia o arquivo para os nós da rede que mantêm cópias do blockchain.
    - Os nós validadores colocam a transação em uma área de espera chamada "mempool" (piscina de memória) para transações válidas, mas não confirmadas.
3. Confirmação (Confirming): Na confirmação o nó principal (e único) validará a transação por meio de verificações seguindo o modelo de UTXO.

5. Resultado Final: LemonBlock

Optamos pela linguagem **Python 3.6** para esse projeto, devido ao escopo conceitual do mesmo e pelas tecnologias que utilizamos, sendo importante citar as bibliotecas **Flask 2.3.3**, para construção de uma interface web, **Pycryptodome 3.18.0** para facilitar as operações envolvendo criptografia RSA e **Virtualenv** para construção da maquina virtual que executa nossa aplicação. Podemos dividir o projeto em 3 partes principais:

1. **LemonBlock** - a blockchain propriamente dita.
2. **JuiceWallet** - a carteira para realizar as operações na LemonBlock transferindo **Lemon** (como batizamos as moedas).
3. **Interface Web** - para interação com a carteira.

5.1. LemonBlock

Sua estrutura está presente no package **blockcahin** do projeto onde as classes mais importantes são:

1. **lemonBlock.py:** Responsável por ditar a estrutura mais básica de cada bloco da nossa blockchain.
2. **merkleTree.py:** Implementa a Merkle Tree responsável por armazenar as transações.
3. **node.py:** implementa o node utilizado pela merkle tree.
4. **networkNodeClient.py:** auxilia no controle da blockchain por parte da rede.

5.2. JuiceWallet

Sua estrutura está presente no package **wallet** do projeto onde as classes mais importantes são:

1. **juiceWallet.py:** Responsável pela construção das carteiras que comportarão nossos lemons.
2. **transaction.py:** Tratas as transações que iram ocorrer no sistema seguindo as regras de UTXO e validações descritas no artigo.
3. **transactionInput:** Representa o input a ser recebido na transação, toda transação é composta por 1 input e 1 output.
4. **transactionOutput.py:** Referente ao output para conclusão da implementação do UTXO na valdação de saldos e transações na blockchain.

```python
class LemonBlock:
    # Constructor for the LemonBlock class
    def __init__(self, time_stamp, transaction_data: dict, previous_block=None):
        self.previous_block = previous_block
        self.transaction_data = transaction_data
        self.timestamp = time_stamp

    # Method to calculate the hash of the block
    def cryptografic_hash(self) -> str:
        block_content = {
            "transaction_data": self.transaction_data,
            "previous_block": self.previous_block.cryptografic_hash() if self.previous_block else None,
            "timestamp": self.timestamp
        }
        return calculate_hash(json.dumps(block_content, indent=None))

    def get_previous_block(self):
        return self.previous_block
```

Figure 3. Estrutura mais básica de um bloco

5.3. Interface Web

Essa seção é referente ao uso do **Flask** e sua estrutura está presente no próprio **app.py** onde tratamos as requests e damos acesso a operações na blockchain por parte do usuário, estando dentro do package **src** tendo como arquivos importantes os templates na pasta **templates** em html e na pasta **static** o css e imagens usados nas páginas. Abaixo veremos o resultado visual do acesso por meio do navegador.

1. **Home** Tela de apresentação do sistema.
2. **Access** Onde é possível acessar carteiras existentes inserindo suas chaves pública e privada, consultar seu saldo e realizar transações de Lemons para outras carteiras por meio do address destas.
3. **Create** Onde é possível criar novas carteiras e receber sua chave pública, privada e address.

6. Conclusões

Devido à natureza desta PoC, podemos concluir que é sim possível utilizar uma blockchain como uma solução segura e rápida para as transações comuns do dia a dia dos servidores, alunos e serviços empregados pela universidade. Porém, é importante salientar que, como a blockchain é uma tecnologia muito diferente do habitual na universidade, talvez leve algum tempo para que todos se adaptem, principalmente para que

todos os serviços ofereçam suporte a ela. Para mais informações sobre o código-fonte e o projeto, acesse o repositório do GitHub.

Segue agora algumas images que mostram o resultado final desta PoC.

Figure 4. Tela Inicial, http://hostname:5000/home

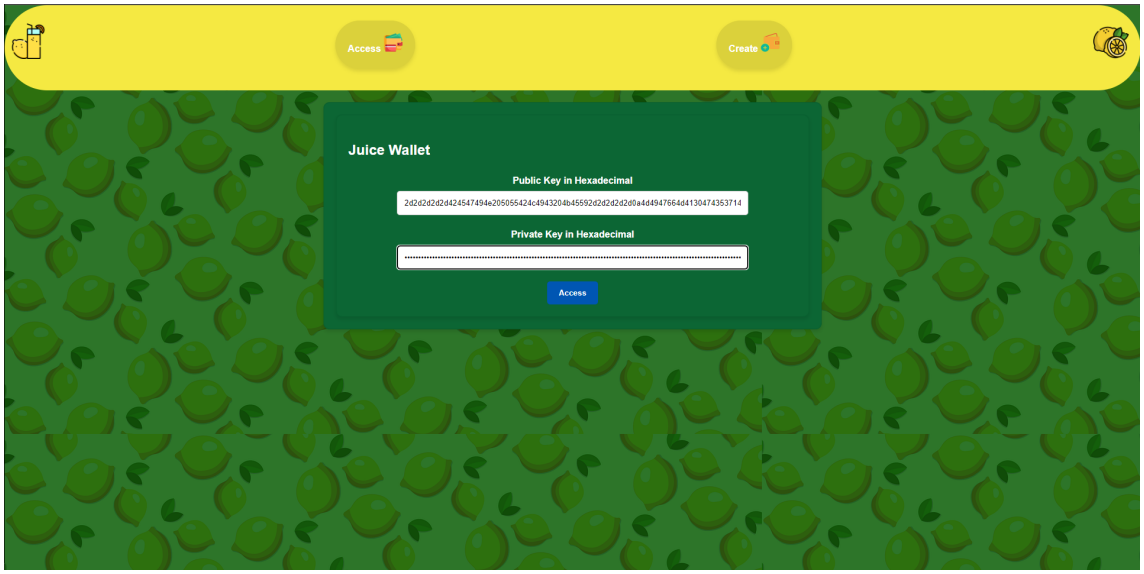

Figure 5. Tela de acesso a carteira, http://hostname:5000/wallet

References

Chausier, G. (2022). Create your own blockchain using python. https://gruyaume.medium.com/create-your-own-blockchain-using-python-4efde6721267. Accessed on 2023-09-12.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*.

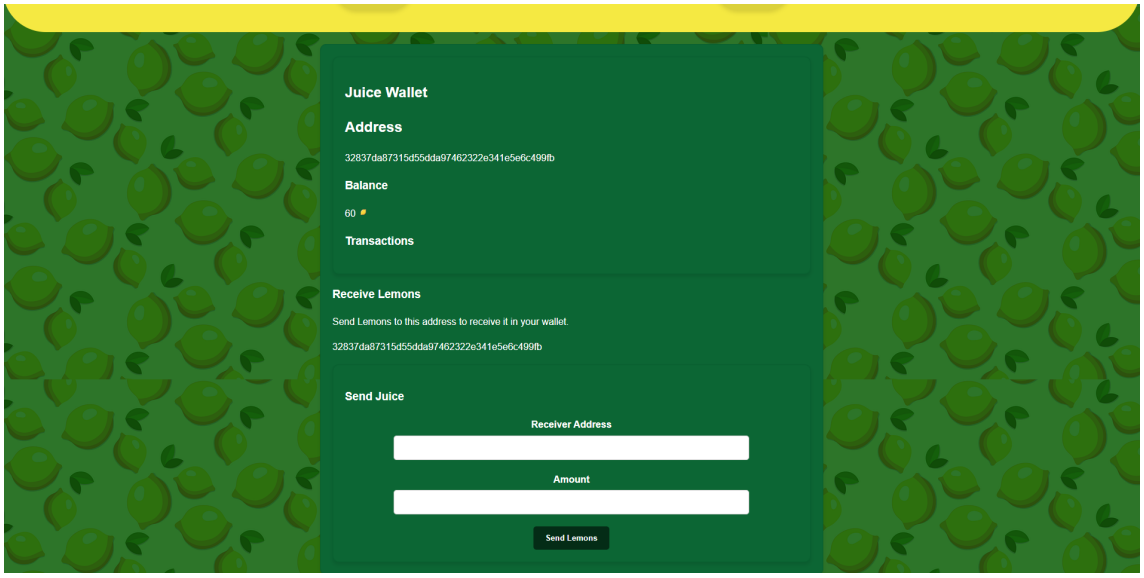

Figure 6. Tela de acesso a carteira, http://hostname:5000/wallet

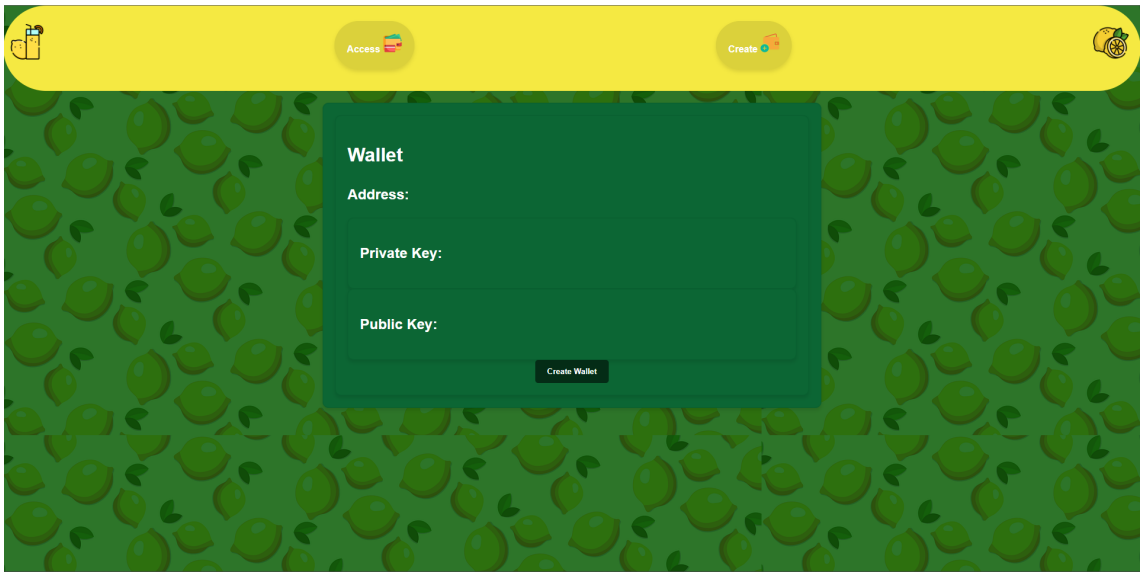

Figure 7. Tela de criação de carteira, http://hostname:5000/wallet-create

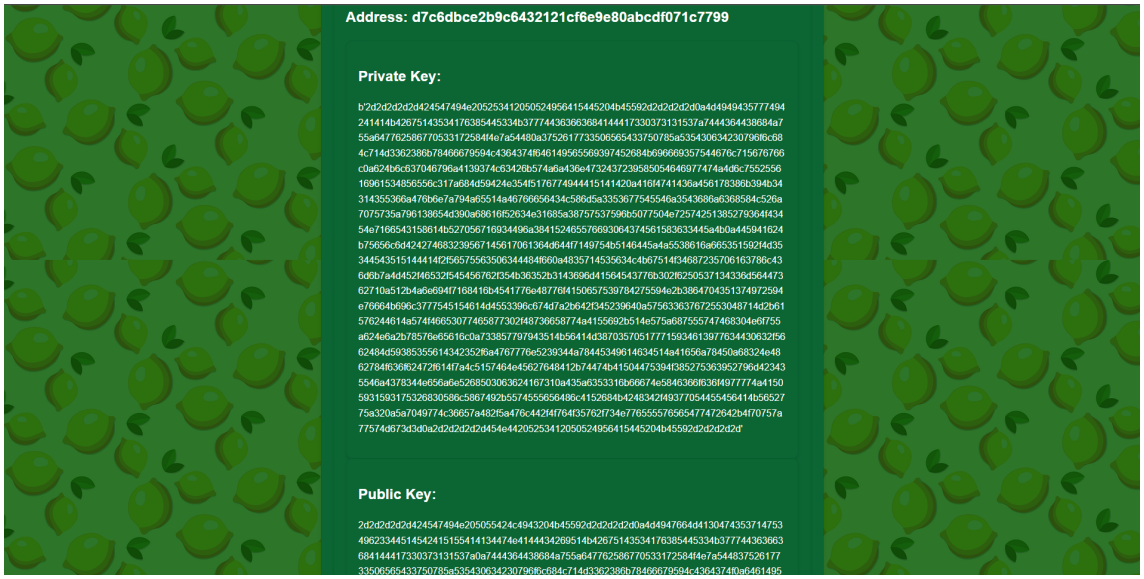

Figure 8. Tela de criação de carteira, http://hostname:5000/wallet-create