

1 2



9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Assignment 2: Blockchain Apps
Todo List - Tuga Coin

Criptografia

Mestrado em Segurança Informática - Departamento de Engenharia Informática

Índice

Índice	1
Contexto	1
TugaCoin	2
Sobre	2
Implementação	2
Tecnologias Usadas	2
Componentes	2
Passo a Passo	3
Todo List App	6
Sobre	6
Smart Contract	6
Implementação	6
Tecnologias Usadas	6
Componentes	6
Passo a Passo	7
Referências	9

Contexto

Este relatório irá incidir essencialmente em dois pequenos projetos, com implementações, funcionalidades e objetivos distintos entre si, porém, ambos com uma temática relacionada com as *blockchains*.

Importante notar que, apesar disso, não são implementações reais! Pelo que servirão apenas para fins didáticos e experimentais.

TugaCoin

Sobre

O *TugaCoin* é, não só uma *blockchain*, implementada de raiz, como também uma criptomoeda.

Implementação

Tecnologias Usadas

- **Node.js** → implementação da *blockchain* e *web server*
- **Bootstrap** → *styling* do *webserver*

Componentes

- Carteira
 - **balanço** → quantia de *TugaCoins* guardados
 - **chave privada** → chave para assinar transações
 - **endereço** → identificador único da carteira
- Transação
 - **endereço remetente** → endereço da carteira que emite a transação
 - **endereço destinatário** → endereço da carteira que recebe a transação
 - **quantidade transacionada** → quantia transacionada
 - **instante** → momento da emissão
 - **quantidade validada** → confirmador da quantia possuída pelo emissor
 - **assina transação()** → autentica a transação com a chave privada da carteira do endereço emissor
- Bloco
 - **hash** → identificador do bloco (resultado do **calcula hash()**)
 - **hash do bloco anterior** → identificador do bloco anterior
 - **instante** → momento da criação do bloco
 - **transações** → transações contidas pelo bloco
 - **nonce** → contador de manipulação da *hash*
 - **minera bloco()** → cria uma *hash*, para o bloco, que tenha, obrigatoriamente, um número de 0s igual à dificuldade definida na *blockchain*, fazendo variar o *nonce*
 - **calcula hash()** → SHA256(previousHash + timestamp + transactions + nonce) a quantidade de transações está implícita na lista de transações, portanto, não precisamos duma variável a mais

- **minera bloco(difficulty)** → encontra uma *hash* que satisfaça o requisito da prova de trabalho imposta
- **tem transações válidas()** → verifica se o bloco apenas contém transações válidas (compre os requisitos referidos anteriormente)

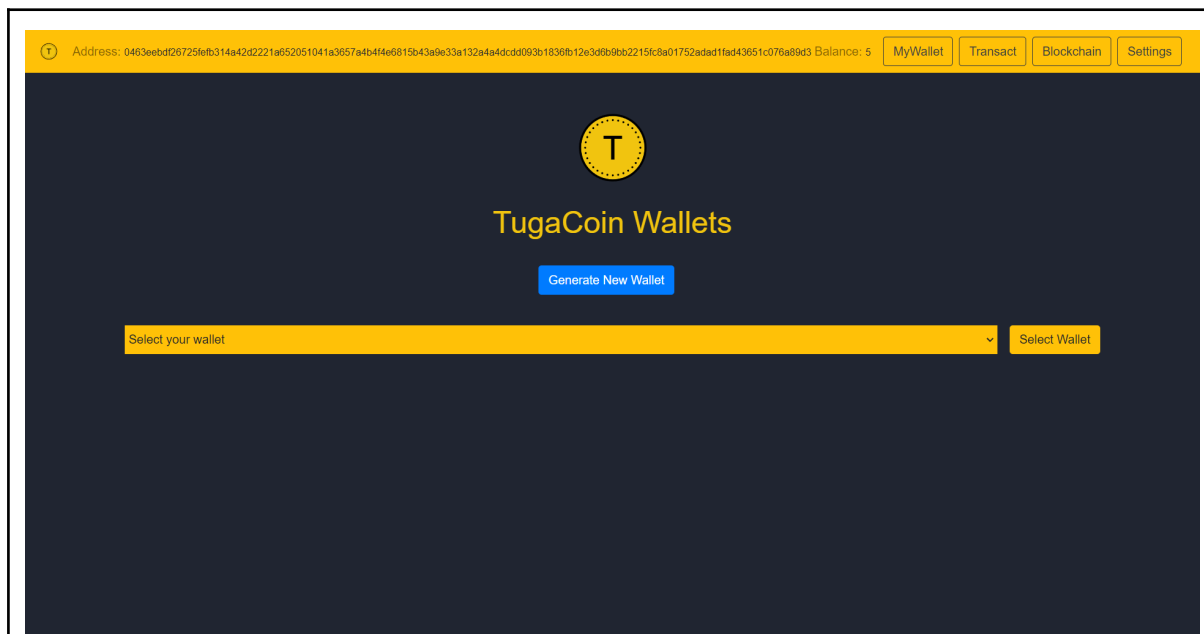
(não foram implementadas as variáveis *magic number*, *version number* e *blocksize* por considerar que não haveria utilidade prática neste contexto)

- **BlockChain**

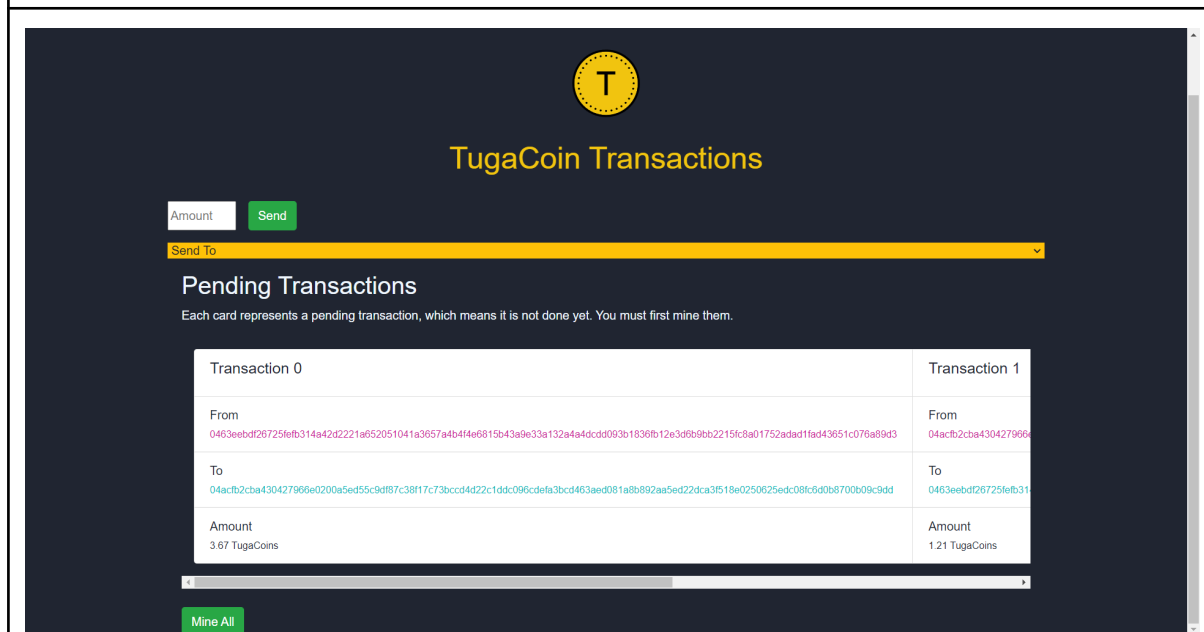
- **cadeia** → lista de blocos
- **dificuldade** → número de 0s iniciais obrigatórios para gerar a *hash* do bloco. Esta variável encontra-se na classe *Blockchain* e não na classe *Block* para que fosse possível manipular a dificuldade da *blockchain* sem causar nenhuma invalidação na geração das *hashes* dos blocos
- **transações pendentes** → transações não confirmadas, pois precisam de ser mineradas
- **recompensa de mineração** → quantia de *TugaCoins* atribuída à carteira que minerar as transações pendentes (alguns cálculos ainda são implementados, veremos quais, no tópico seguinte)
- **a cadeia é válida()** → percorre todos os blocos, verificando se a *hash* de cada bloco e a *hash* do bloco anterior são íntegras & verifica se cada transação foi devidamente assinada pela chave privada duma carteira
- **cria bloco gênese()** → cria o primeiro bloco da cadeia
- **minera transações pendentes(minerador, carteiras)** → O endereço passado concretiza a prova de trabalho em todas as transações pendentes. Ele recebe a recompensa e as transações entre carteiras são efetuadas
- **a cadeia é válida()** → Verifica se: 1- os blocos apenas contém transações válidas, 2- a *hash* de cada bloco foi bem gerada/é íntegra, de acordo com os dados que armazena, 3- a *hash* do bloco anterior é corresponde à ***previousHash*** do bloco atual

Passo a Passo

Ao gerar, e selecionar, uma carteira vemos que já temos 5 *TugaCoins* disponíveis para transferência. Vemos que temos também 4 abas disponíveis para acesso. Tendo em conta que a opção *MyWallet* é aquela na qual estamos atualmente, acedamos à opção *Transact*.



Ao definirmos uma carteira destino e uma quantia, a transação é criada, porém, ainda não é, de facto, feita, pois ainda precisa de ser minerada. De notar que, nesta altura, os 5 *TugaCoins* ainda não são descontados da carteira do utilizador.



Se, com a mesma carteira, minerarmos todas as transações pendentes, um bloco é criado e pode ser consultado na aba *Blockchain*.

Aqui, podemos ver que um bloco foi criado (o bloco 0 é o bloco génese, que, na prática, serve apenas para dar origem à cadeia de blocos) e que, nele, estão contidas todas as transações que estavam, até então, pendentes.

Questão: O que aconteceria se, com os 5 *TugaCoins* iniciais na carteira, fizesse uma transação de 3 *TugaCoins*, seguida de outra de 4 *TugaCoins*?

Nesta situação, ambas as transações são criadas, uma vez que, enquanto elas estiverem pendentes, o balanço da carteira do utilizador não é alterado. O que acontece, na verdade, é que, só durante a mineração do bloco é que as alterações entre carteiras são


feitas, e, caso uma carteira remetente não possua balanço suficiente relativamente à quantia transferida, a transação adquire o estado de *amountValidated=false*, o que se refletiria na não realização da transferência, o que seria visível na coluna *Valid* abaixo, com um X.

Block 0 (Genesis block)		Block 1	
Hash	92f0d54e0d1af3bfa68b792b5300cd98c797d0737327d9fe6891905b992eb2ca	Hash	00ce74bad59c311782eeff0ccbe2d9d1cee702a7ccc09ea782e7b1989739308
Hash of previous block	0	Hash of previous block	92f0d54e0d1af3bfa68b792b5300cd98c797d0737327d9fe6891905b992eb2ca
Nonce	0	Nonce	161
Timestamp	Jan 1, 2017, 24:00	Timestamp	Jan 1, 2022, 23:27
		View Transactions	

#	From	To	Amount	Timestamp	Valid
0	System	0463eebdf26725feb314a42d221a652051041a3657a4b44e...	20 (Block reward)	1641079641689 Jan 1, 2022, 23:27	✓
1	0463eebdf26725feb314a42d221a652051041a3657a4b44e...	04acfb2cba430427966e0200a5ed55c9d87c38f17c73bccd4d2...	3.67	1641079565140 Jan 1, 2022, 23:26	✓
2	04acfb2cba430427966e0200a5ed55c9d87c38f17c73bccd4d2...	0463eebdf26725feb314a42d221a652051041a3657a4b44e...	1.21	1641079586993 Jan 1, 2022, 23:26	✓


Podemos ainda aceder às configurações da *blockchain* para alterar a dificuldade e a recompensa da mineração. De notar que a recompensa efetivamente atribuída é o resultado da multiplicação destas duas variáveis por bloco, ou seja:

Recompensa Final = dificuldade * recompensa mineração * transações pendentes
 ex: Recompensa Final = 2 * 5 = 10 *TugaCoins* / por transação


 Address:
 0463eebdf26725feb314a42d221a652051041a3657a4b44e6815b43a9e33a132a4a4dcd0903b1836fb12e3d8b9bb2215fc8a01752adad1fa43651c076a89d3

Balance:
 22.54

[MyWallet](#)
[Transact](#)
[Blockchain](#)
[Settings](#)



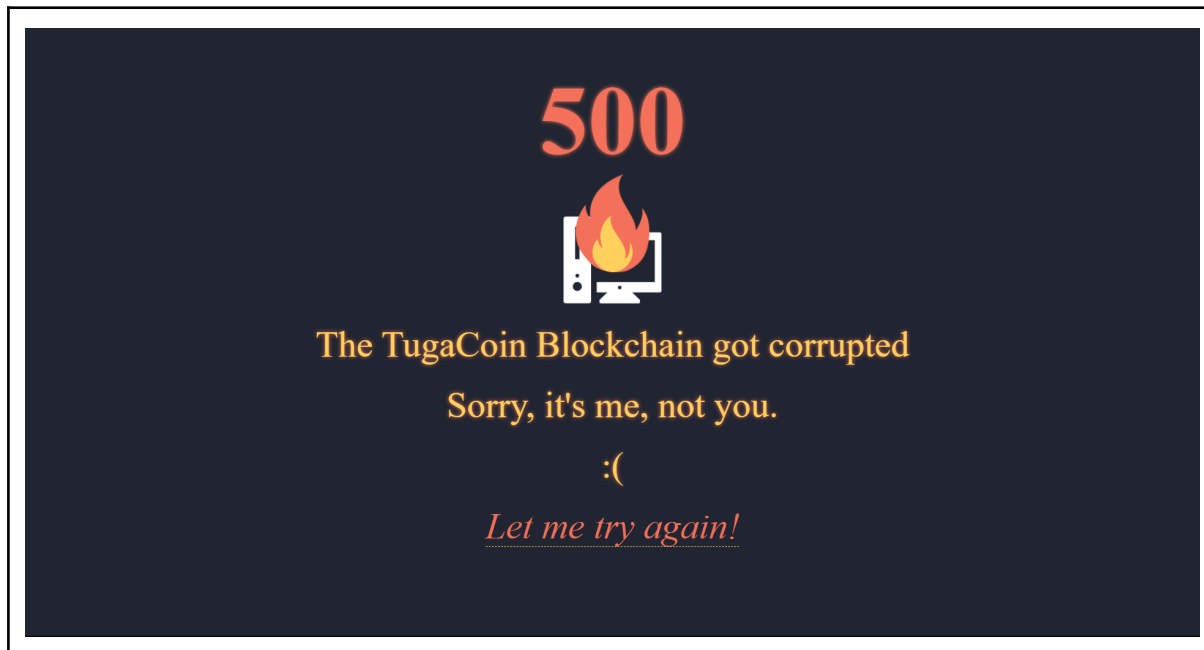
TugaCoin Settings

Difficulty:

Mining Reward:

[Submit Changes](#)

Se, por alguma razão, a função **a cadeia é válida()** retornar *false*, significa que a *blockchain* já não é íntegra e, como um dos princípios primordiais desta tecnologia é a integridade dos seus dados, nesta situação, em específico, esta página é apresentada ao utilizador e a *blockchain* é reiniciada.



Todo List App

Sobre

Esta é uma aplicação *web*, onde, através duma *blockchain* e de tecnologia baseada na *Ethereum*, a lista de tarefas é representada por um *smart contract*.

Smart Contract

Um *smart contract* é, de forma metafórica, como uma angariação ou promessa de que algo vai acontecer por meio de uma ou várias transações, feitas através duma *blockchain*, o que confere a esta tecnologia garantias de imutabilidade e distribuição.

Ou seja, usando este contexto como base de exemplo, ao criar, editar, eliminar ou concluir uma tarefa, é feita uma transação duma certa quantidade de *ETH* como garantia de que a alteração vai ser, efetivamente feita. Caso, por alguma razão, a quantia requerida não seja suficiente para efetuar a alteração (o que, neste contexto, não acontece), todas as transações angariadas até então serão retribuídas de volta às carteiras origem.

Implementação

Tecnologias Usadas

- **Node.js** → implementação do *web server*
- **Bootstrap** → *styling* do *webserver*
- **Ganache** → criação duma *blockchain Ethereum* pré montada e automatizada local
- **Extensão Web MetaMask** → ponte entre o *web client* e a *blockchain*

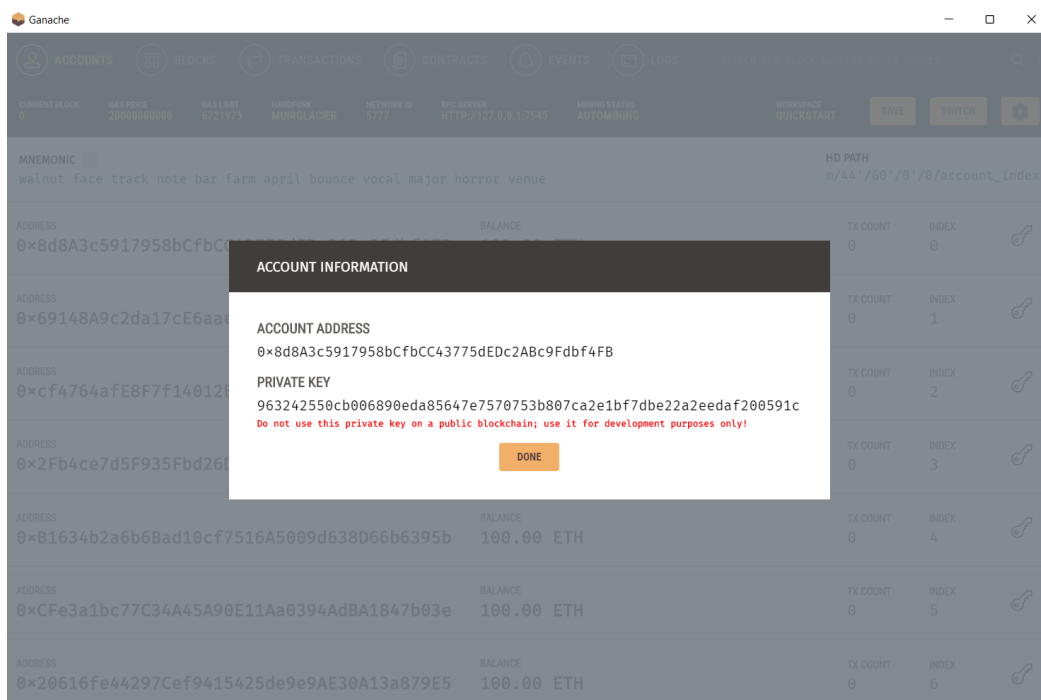
- **Solidity** → criação de *smart contracts* baseados na *blockchain* da *Ethereum*

Componentes

- Todo List - *Smart Contract*
 - **tasks** (mapping → Task)
 - id → identificador único da tarefa
 - descrição → descrição da tarefa
 - data conclusão → data de conclusão da tarefa
 - completada → estado da tarefa
 - **eventos**
 - tarefaCriada
 - tarefaApagada
 - tarefaEditada
 - tarefaEstadoAlterado

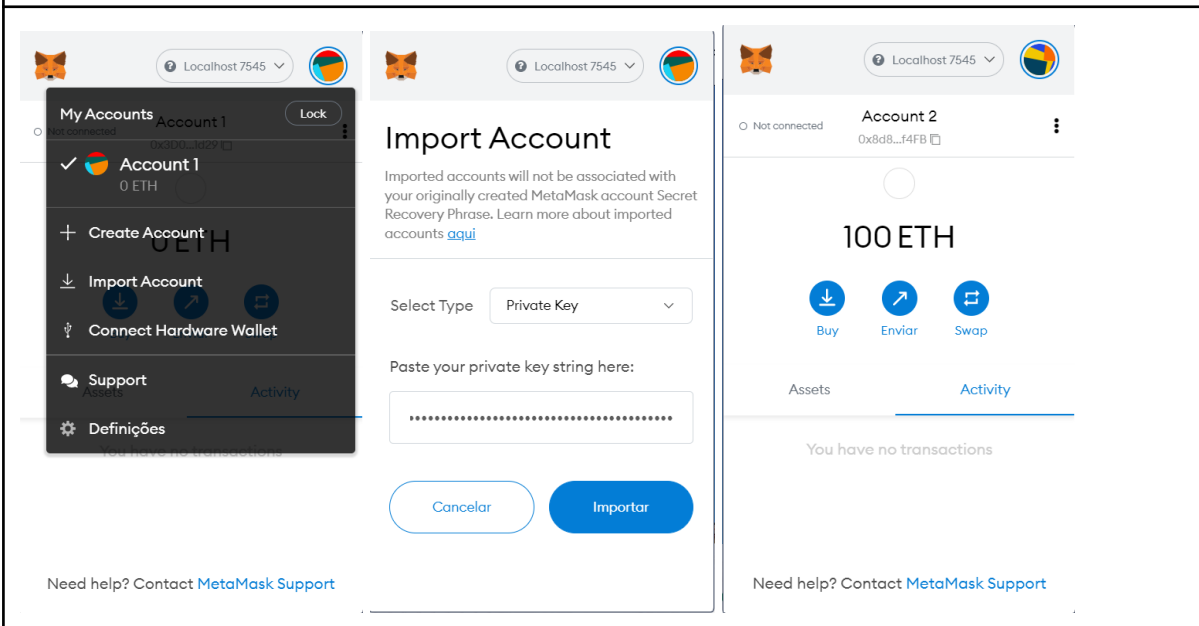
Passo a Passo

1. Instalar Ganache
 - 1.1. Selecionar opção *QuickStart*
 - 1.2. Copiar chave privada do primeiro endereço disponível na *blockchain* apresentada

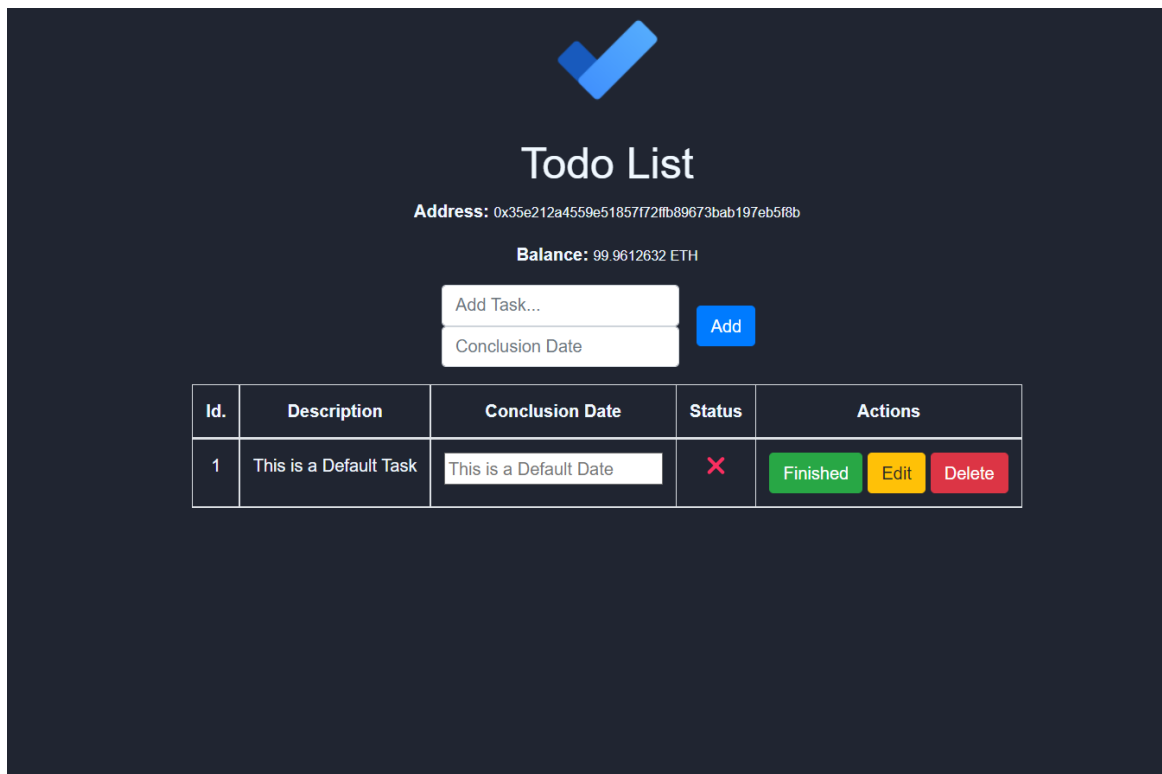


2. Instalar MetaMask

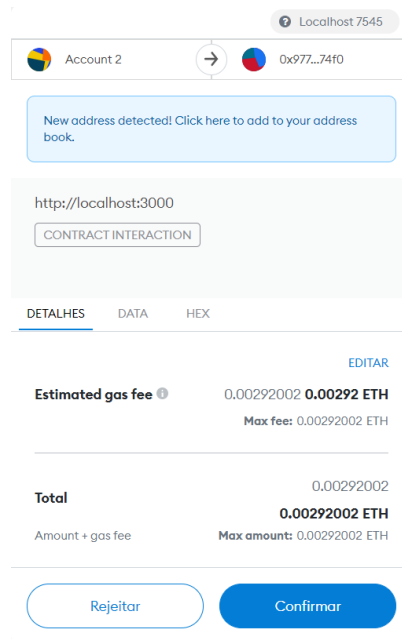
- 2.1. Criar Conta/Carteira Default (não necessária para este processo)
- 2.2. Configurar opções de rede para ser conectar ao *Ganache* (IP: 127.0.0.1 Porto: 7545)
- 2.3. Importar Carteira
- 2.4. Colar chave privada



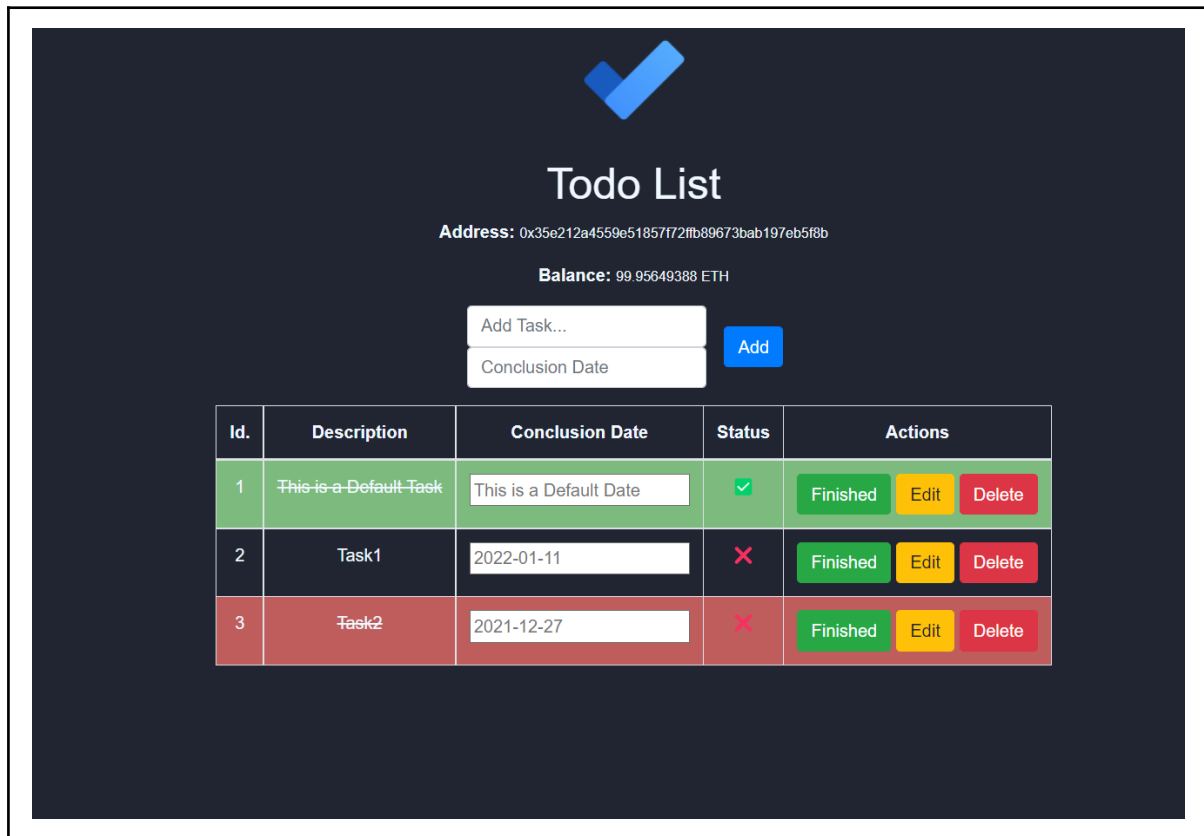
3. Na diretoria do projeto (*ETH_Blockchain_Todo*) executar:
 - 3.1. `npm install -g truffle@5.0.2`
 - 3.2. `npm install`
 - 3.3. `truffle compile`
 - 3.4. `truffle migrate --reset`
 - 3.5. `npm run start`
4. O seguinte menu será apresentado e pode:
 - 4.1. Consultar o endereço e balanço da sua carteira
 - 4.2. As suas tarefas
 - 4.3. Criar novas tarefas
 - 4.4. Editar a descrição e data de cada tarefa (escrevendo novos conteúdos na célula da tabela correspondente)
 - 4.5. Eliminar tarefas



5. Ao, por exemplo, criar uma tarefa é pedida uma transação, como referido no tópico [Smart Contract](#). Este processo é igual para qualquer outra alteração.



6. Depois de algumas alterações podemos ver também que:
- 6.1. A tarefa concluída ficou verde
 - 6.2. A tarefa cuja data de conclusão já passou ficou vermelha
 - 6.3. O balanço foi alterado



Referências

- *Slides teóricos da cadeira*