

Proyecto 1 - Aurax: Documento explicativo

Universidad Central de Venezuela
Desarrollo de Aplicaciones Distribuidas
Felix G. Urbano U.
V-25369306

Abstract. La creación de la tecnología blockchain y las criptomonedas ha sido una de las invenciones más disruptivas e importantes en la actualidad. Las capacidades que ofrece dicha tecnología como base de datos distribuida para apoyar a procesos de negocios existentes y creación de nuevos mercados ha conllevado a los profesionales de la tecnología entender su funcionamiento así como las consideraciones de su implementación. Este artículo presenta de forma resumida el funcionamiento, uso y puntos clave de la tecnología blockchain bajo un enfoque práctico desarrollando un sistema de transferencia de dinero digital a modo de introducción al mundo de las criptomonedas.

Introducción

El comercio por internet ha tenido siempre una fuerte dependencia de las instituciones financieras que actúan como un tercero para verificar, procesar y validar los pagos electrónicos, esto principalmente consecuencia del problema de doble gasto.

En un entorno real del mundo físico si una entidad A desea enviar una determinada cantidad de dinero a una entidad B, una vez que se realice la transacción la entidad A dejaría de tener en su posesión el dinero enviado ya que lo tendría la entidad B. Sin embargo en un entorno virtual el problema de doble gasto se presenta ya que el dinero está en forma digital y se puede duplicar fácilmente. En este punto es donde las instituciones financieras entran como el responsable centralizado de procesar, validar y verificar las transacciones que se realizan entre los pares involucrados.

La tecnología blockchain ofrece la posibilidad de descentralizar el procesamiento y validación de las transacciones realizadas entre un par de entidades, funcionando como un libro público y contable que hace seguimiento de las transacciones generadas.

Para poder ejecutar las tareas, la blockchain debe cumplir con las características mencionadas a continuación

- **Distribuido:** El libro contable (blockchain) debe de replicarse en varias computadoras o nodos (se encuentran conectados entre sí) que procesan y validan las transacciones realizadas además de que cualquiera de estos nodos puede descargar una copia completa de la blockchain en cuestión.
- **Criptográfico:** Debe usarse un mecanismo de criptografía para asegurar o autorizar transacciones. También permite el anonimato entre pares al registrar las transacciones en el bloque
- **Inmutable:** La blockchain solo permite añadir nuevos bloques a la cadena, no se pueden eliminar ni modificar.
- **Consenso:** Debe implementar un mecanismo que permita establecer un acuerdo entre todos los nodos de la sobre la información que manejan, garantizando una única fuente de verdad. Aunque existen diversos algoritmos de consenso para este fin, nos enfocaremos en la prueba de trabajo o Proof of work donde un tipo especial de participantes en la red llamados mineros compiten en la búsqueda de la solución a un rompecabezas criptográfico que les permitirá agregar un bloque de transacciones a la cadena de bloques. Añadiendo así seguridad al sistema.

Transacciones

A continuación se explicará de forma resumida el flujo necesario para poder realizar una transacción .

Para poder operar con una criptomoneda todo usuario necesita crear una billetera o wallet, la misma almacena 2 cadenas de caracteres denominadas llaves, las cuales pueden ser públicas o privadas.

La llave pública sirve como dirección o referencia de quien recibe el activo en cuestión, está debe ser compartida entre los pares al momento de realizar una transacción. Es importante entender que las billeteras no almacenan los activos en sí, ya que los saldos se calculan a partir de las transacciones registradas en la blockchain.

Por otra parte, la llave privada es secreta y no debe ser compartida. Sirve como un mecanismo de autorización o firma que permite al usuario realizar una transacción.

Creada la billetera, se procede a adquirir los activos o la criptomoneda de la blockchain a través de una compra de éstos. Una vez con saldo, se puede proceder a realizar una transacción. Para esto el usuario emisor necesitará conocer la llave pública o dirección del usuario receptor de la transacción.

Conociendo la llave pública del receptor, el emisor puede proceder a realizar la transacción definiendo la cantidad a enviar y usando su llave privada como autorización.

La transacción se transmite a toda la blockchain, es decir a todos los nodos conectados a dicha red de blockchain. Una vez se recibe la transacción cada nodo competirá por escribir el siguiente bloque de la blockchain, esto lo logran resolviendo un rompecabezas criptográfico a fuerza bruta, la respuesta a este rompecabezas se le conoce como "nonce".

El nodo que resuelve el rompecabezas, transmite el cambio de la blockchain a todos sus pares, para que confirmen la transacción y que si es válido cada uno de los nodos de la red tengan sus copias de la cadena sincronizadas.

Prueba de trabajo

La prueba de trabajo es un algoritmo de consenso que se usa para evitar acciones no deseadas sobre la blockchain. Consiste en realizar un trabajo complejo de operaciones computacionales (en este caso encontrar la solución del rompecabezas criptográfico) que posteriormente será verificado por el resto de la red o nodos participantes. Realizado el trabajo el nodo recibe un incentivo o recompensa.

Incentivo

La primera transacción registrada en un bloque asigna una ganancia (una cierta cantidad de la criptomoneda de la blockchain) como recompensa por crear dicho bloque.

Así mismo sucede con cualquier otro bloque que sea validado y añadido a la cadena, el nodo que lo añade recibe la recompensa establecida.

La recompensa es un aspecto importante a resaltar ya que el proceso de minado o resolución del rompecabezas criptográfico requiere un consumo considerable de CPU y electricidad además que dicha recompensa es atractiva económicamente y logra atraer a nuevos mineros a la red.

Aurax

A efectos prácticos para explicar el funcionamiento de la tecnología blockchain y su uso en concreto como pilar de las criptomonedas, se decidió crear una criptomoneda llamada Aurax que existe dentro de una blockchain sencilla ya existente implementada con Python y Flask.

Aurax sirve como un activo digital de intercambio entre pares que permite introducir a principiantes a este sistema de intercambio de activos.



Imagen 1. Logo de la criptomoneda Aurax

Implementación

Como se mencionó anteriormente la blockchain utilizada para el proyecto ya existía y se extrajo del ejemplo presentado en el blog de A. Moujahid: A Practical Introduction to Blockchain with Python

Sin embargo se realizaron modificaciones sobre la misma para ejecutar el cálculo del balance de una billetera dada su clave pública.

Por otra parte, para interactuar con dicha blockchain se creó un API REST que sirve como sistema de intercambio de activos y trabaja en conjunto con un cliente desarrollado en Angular.

Para desplegar los tres proyectos involucrados se usó la tecnología de contenedores de Docker en conjunto con docker-compose de forma tal que a través de un solo comando puedan desplegarse los contenedores de las aplicaciones involucradas ya que cada uno consta de su propio archivo Dockerfile para generar la imagen correspondiente.

Entre las características de la blockchain podemos resaltar:

- Posibilidad de agregar múltiples nodos a la cadena de bloques
- Prueba de trabajo (PoW)
- Resolución simple de conflictos entre nodos
- Transacciones con cifrado RSA

A nivel del API REST como sistema de intercambio de activos:

- Generación de billeteras usando cifrado de clave pública/privada (basado en el algoritmo RSA).
- Registro y autenticación en el sistema
- Autorización usando JWT

- Generación de transacciones con encriptación RSA
- Adquisición y retiro de Tokens
- Intercambio de activos a Tokens, Aurax y Dólares

Recalcando que los procesos de compra, venta e intercambio no tienen un valor real ni existe ningún tipo de transacción a través de una pasarela de pago ni de terceros, ya que no corresponde al alcance ni al objetivo del proyecto.

Finalmente, el cliente desarrollado en angular presenta la siguientes características:

- Pantalla de registro, autenticación y autorización en el sistema
- Pantalla para compra de tokens a través de dólares
- Pantalla para ejecutar intercambios de los activos:
 - Token a Dólar
 - Token a Aurax
 - Aurax a Token
 - Aurax a Dólar

La estructura del proyecto se establece de la siguiente manera, tal como se aprecia en la Imágen 2

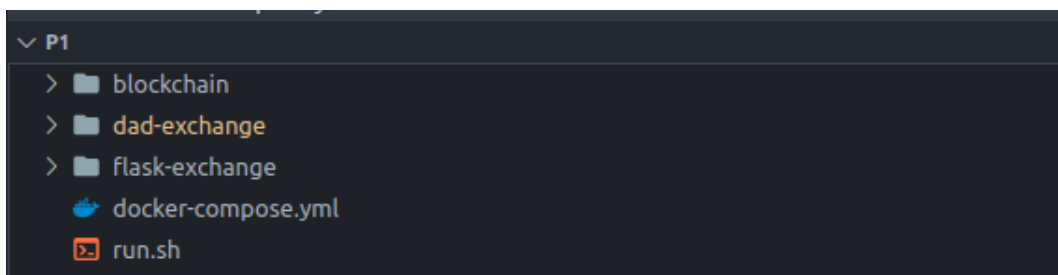


Imagen 2. Estructura del proyecto

- **blockchain:** Es el proyecto que define la estructura y funcionamiento de la blockchain. Se ejecuta en el puerto 5000
- **flask-exchange:** Es el API REST para realizar todas las transacciones sobre la blockchain usando Aurax como moneda así como las conversiones a Dólares, Tokens y Aurax solicitadas por los usuarios. Se ejecuta en el puerto 5001
- **dad-exchange:** Es el cliente en angular que sirve de interfaz de usuario para interactuar con la plataforma de intercambio de activos. Se ejecuta en el puerto 4200

El archivo docker-compose.yml se encarga de crear y desplegar los contenedores de los proyectos anteriormente mencionados.

```

docker-compose.yml
You, hace 2 semanas | 1 author (You)
1  version: "3.9"
2  services:
3
4      blockchain:
5          build:
6              context: ./blockchain
7          image: blockchain
8          ports:
9              - "5000:5000"
10         container_name: blockchain
11         networks:
12             vpcbr:
13                 ipv4_address: 10.5.0.7
14
15     backend:
16         build:
17             context: ./flask-exchange
18         image: backend
19         ports:
20             - "5001:5001"
21         container_name: backend
22         networks:
23             vpcbr:
24                 ipv4_address: 10.5.0.5
25
26     fronted:
27         build:
28             context: ./dad-exchange
29         image: frontend
30         ports:
31             - "4200:80"
32         container_name: frontend
33         networks:
34             vpcbr:
35                 ipv4_address: 10.5.0.6
36
37     networks:
38         vpcbr:
39             driver: bridge
40             ipam:
41                 config:
42                     - subnet: 10.5.0.0/16
43                       gateway: 10.5.0.1
44

```

Imagen 3. Extracto del archivo docker-compose.yml

Despliegue

Para facilitar el proceso de creación y borrado de imágenes docker y sus respectivos contenedores se creó un archivo *run.sh* para tal fin.

El mismo puede ser ejecutado en Linux con el comando **sh run.sh**, tal como se aprecia en la siguiente imagen.

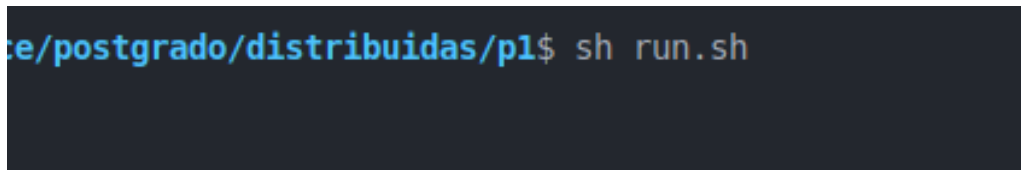


Imagen 4. Comando para la ejecución del proyecto

Blockchain

El proyecto basado en una arquitectura monolítica presenta una interfaz desarrollada usando el motor de plantillas de Flask jinja2.

Presenta dos pestañas en la barra de navegación, tal como podemos ver en la siguiente imagen:

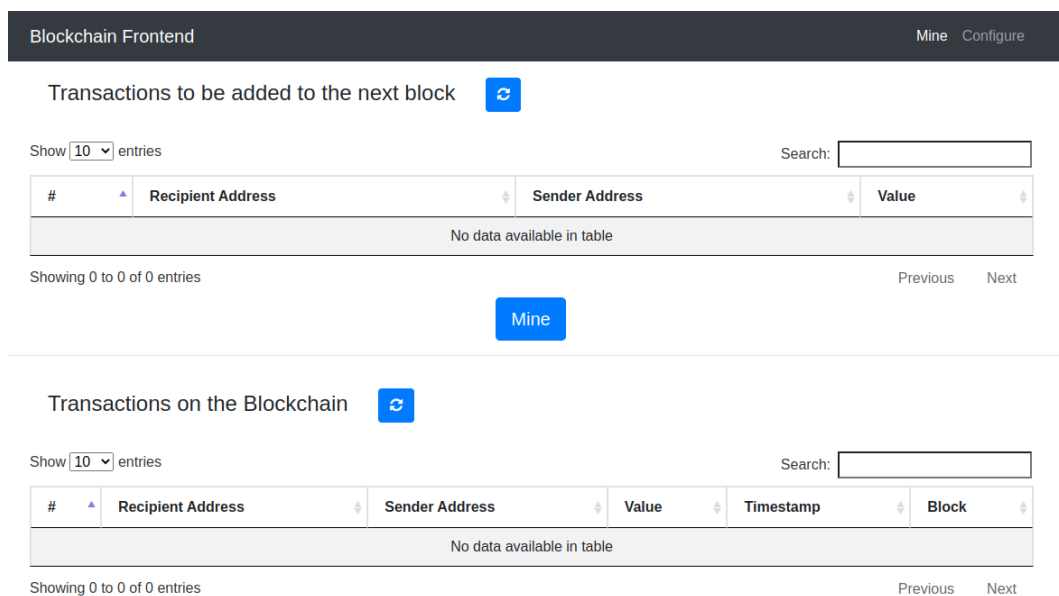


Imagen 5. Tablero principal del proyecto blockchain

- Mine: para ver transacciones y datos de blockchain, y para extraer nuevos bloques de transacciones.
- Configurar: Para configurar las conexiones entre los diferentes nodos de la cadena de bloques.

A continuación se presentará una breve explicación de las partes más importantes éste proyecto.

Se implementó una clase Blockchain que cuenta con los siguientes atributos

- **transactions:** Lista de transacciones que se agregarán al siguiente bloque.
- **chain:** La cadena de bloques real, que es una matriz de bloques.
- **nodes:** Un arreglo de string contiene direcciones URL de nodos. La blockchain usa estos nodos para recuperar datos de la blockchain de otros nodos y de esta forma sincronizarse.
- **node_id:** Sirve como identificador del nodo en cuestión

Así mismo implementa los siguientes métodos

- **register_node(node_url):** agrega un nuevo nodo de cadena de bloques a la lista de nodos.
- **verify_transaction_signature(sender_address, signature, transaction):** Comprueba que la firma proporcionada corresponde a la transacción firmada por la clave pública (sender_address).
- **submit_transaction(sender_address, recipient_address, value, signature):** agrega una transacción a la lista de transacciones si se verifica la firma.
- **create_block(nonce, previous_hash):** Agrega un bloque de transacciones a la cadena de bloques.
- **hash(block):** crea un hash SHA-256 de un bloque.
- **proof_of_work():** Algoritmo de prueba de trabajo. Busca un nonce que satisfaga la condición de minería.
- **valid_proof(transactions, last_hash, nonce, difficulty=MINING_DIFFICULTY):** Comprueba si un valor hash satisface las condiciones de minería. Esta función se utiliza dentro de la función proof_of_work.
- **valid_chain(chain):** comprueba si una cadena de bloques es válida.
- **resolve_conflicts():** Resuelve conflictos entre los nodos de blockchain al reemplazar una cadena con la más larga de la red

Endpoints

El proyecto base establece los siguientes endpoints, los cual podemos agruparlos según la funcionalidad que exponen

Transacciones:

- **'/transactions/new':** Esta API toma como entrada 'sender_address', y 'recipient_address', y agrega la transacción a la lista de transacciones que se agregarán al siguiente bloque si la firma es válida.
- **'/transactions/get':** Esta API devuelve todas las transacciones que se agregarán al siguiente bloque.
- **'/chain':** Esta API devuelve todos los datos de la cadena de bloques.
- **'/mine':** esta API ejecuta el algoritmo de prueba de trabajo y agrega el nuevo bloque de transacciones a la cadena de bloques.

- **'/balance/<pubkey>'**: Obtiene la información del balance de una cuenta, con esta información se determina si una operación puede realizarse siempre que se cuenten con los fondos disponibles.

Configuración

- **'/nodes/register'**: esta API toma como entrada una lista de direcciones URL de nodos y las agrega a la lista de nodos.
- **'/nodes/resolve'**: Esta API resuelve conflictos entre nodos de blockchain reemplazando una cadena local con la más larga disponible en la red.
- **'/nodes/get'**: Esta API devuelve la lista de nodos.

```
@app.route('/transactions/new', methods=['POST'])
> def new_transaction(): ...

@app.route('/transactions/get', methods=['GET'])
> def get_transactions(): ...

@app.route('/balance/<pubkey>', methods=['GET'])
> def get_balance(pubkey): ...

@app.route('/chain', methods=['GET'])
> def full_chain(): ...

@app.route('/mine', methods=['GET'])
> def mine(): ...

@app.route('/nodes/register', methods=['POST'])
> def register_nodes(): ...

@app.route('/nodes/resolve', methods=['GET'])
> def consensus(): ...

@app.route('/nodes/get', methods=['GET'])
> def get_nodes(): ...
```

Imagen 6. Extracto de los endpoints del proyecto de blockchain

Cálculo de balance

Para determinar el balance de una billetera el proceso que se estableció fue de recorrer la cadena completa calculando el total de los montos enviados y los montos recibidos a la dirección de la clave pública suministrada.

En la siguiente imagen podemos observar el proceso para calcular la cantidad de montos enviados, esto es sumar los montos enviados ya incorporados al bloque y los no incorporados al bloque.

```
@app.route('/balance/<pubkey>', methods=['GET'])
def get_balance(pubkey):
    public_key = str(pubkey)
    # Se recorre la cadena y se obtienen los montos enviados a la dirección pública
    transactions_sent = [
        [float(tx['value']) for tx in block['transactions']
         if tx['sender_address'] == public_key]
        for block in blockchain.chain]
    # Se obtienen los montos enviados que aun no se han incorporado al bloque
    open_transactions = [
        float(
            tx['value']) for tx in blockchain.transactions
        if tx['sender_address'] == public_key]
    transactions_sent.append(open_transactions)
```

Imagen 7. Extracto del código para el cálculo del balance de una billetera, montos enviados desde la billetera

Una vez que se obtiene el total enviado desde esa billetera se calcula el monto total recibido.

```
# Se recorre la cadena y se obtienen los montos recibidos a la dirección pública
transactions_received = [
    [float(tx['value']) for tx in block['transactions']
     if tx['recipient_address'] == public_key]
    for block in blockchain.chain]
# Con todos los montos de transacciones recibidas procedemos calcular
# el monto total recibido
amount_received = reduce(
    lambda tx_sum, tx_amt: tx_sum + sum(tx_amt)
    if len(tx_amt) > 0 else tx_sum + 0, transactions_received, 0)

balance = amount_received - amount_sent
return jsonify(json.dumps({'balance': balance,
    'amount_received': amount_received,
    'amount_sent': amount_sent})), 200
```

Imagen 8. Extracto del código para el cálculo del balance de una billetera, montos recibidos a la billetera

Finalmente, el balance de la billetera se obtiene como la diferencia del total de dinero recibido y el total de dinero enviado.

Sistema de intercambio de activos (Exchange)

El API REST establece los siguientes endpoints

- **'/exchange-rate'** : Retorna la tasa de intercambio y valor de los activos.
- **'/exchange'**: Ejecuta la conversión de los activos a intercambiar con el sistema
- **'/sign-in'** : Permite la autenticación de usuarios al sistema y retorna el token JWT en caso de ser exitoso.
- **'/sign-up'**: Permite el registro de nuevos usuarios al sistema
- **'/beneficiaries'**: Retorna la lista de los beneficiarios registrados en el sistema, así como su llave pública para facilitar el proceso de transferencia
- **'/tokens'**: Permite la compra de tokens del usuario contra el sistema
- **'/wallet/<public_key>/tokens'**: Retorna los tokens asociados a la llave pública suministrada
- **'/balance'**: Retorna el balance de la billetera de Aurax como de tokens
- **'/transfer'**: Permite realizar la transferencia de Aurax entre usuarios, invoca al endpoint /transactions/new de la cadena de bloques

Tokens

Los tokens son el activo base que permiten participar en el sistema de intercambio, ya que para poder realizar transacciones con AURAX (la moneda de la blockchain) primero deben comprarse token que en un futuro serán intercambiados por AURAX para poder realizar las transacciones con otros usuarios.

Sin embargo, la compra de tokens no son registradas dentro de la blockchain en este caso sino que se almacenan como un valor dentro de un archivo de texto.

Una vez que un usuario tenga tokens en su control podrá cambiarlos por AURAX o retirarlos como Dólares.

Intercambio de activos

Se estableció la siguiente correspondencia entre los activos utilizados en la plataforma (Tokens, Aurax y Dólares)

AURAX	TOKEN	DÓLAR
0.001	1	10

Tabla 1. Correspondencia de valor entre activos

El cálculo de conversión sigue la siguiente fórmula: $VC = M * DM / SM$

Donde:

- VC : Monto a recibir por la conversión
- M: Monto a cambiar
- DM: Tasa de cambio del activo destino
- SM: Tasa de cambio del activo origen

La siguiente imagen presenta el segmento de código encargado de realizar el intercambio de activos contra el sistema

```
@app.route('/exchange', methods=['POST'])
@jwt_required()
def exchange_currency():
    form = request.json
    private_key = get_wallet_private_key(get_jwt_identity())

    from_currency = EXCHANGE_RATE[form['to_currency']]
    to_currency = EXCHANGE_RATE[form['from_currency']]
    exchange_amount = form['amount'] * from_currency / to_currency

    try:
        if (form['from_currency'] == 'token'):
            if (form['to_currency'] == 'dollar'):
                manage_tokens(form['token_wallet'], form['amount'], 'remove')
            else: # TOKENS --> AURAX
                # actualizar archivo con tokens -= new_tokens
                manage_tokens(form['token_wallet'], form['amount'], 'remove')
                # Transferir del sistema al usuario
                transfer_to_user(
                    form['node'],
                    SYSTEM['wallet']['public_key'],
                    SYSTEM['wallet']['private_key'],
                    form['coin_wallet'],
                    str(exchange_amount),
                    True)
        else:
            if (form['to_currency'] == 'token'):
                # Transferir de cuenta usuario a cuenta sistema
                transfer_to_user(
                    form['node'],
                    form['coin_wallet'],
                    private_key,
                    SYSTEM['wallet']['public_key'],
                    str(form['amount']))
                manage_tokens(form['token_wallet'], exchange_amount, 'add')
            else:
                transfer_to_user(
                    form['node'],
                    form['coin_wallet'],
                    private_key,
                    SYSTEM['wallet']['public_key'],
                    str(form['amount']))
    )
    return jsonify({'data': 'Transacción exitosa'}), 200
except Exception as e:
    print(e)
    print(traceback.format_exc())
    return jsonify({'data': 'Ha ocurrido un error'}), 500
```

Imagen 9. Extracto del código encargado de realizar la conversión de activos

Envío de transacciones AURAX

Una vez que se realice la solicitud de una transacción en AURAX desde el sistema de intercambio de activos se realiza una petición al nodo que se defina. las transacciones sólo será efectiva una vez que el nodo mine la transacción. Para esto deberá acceder a la url del nodo en cuestión y minar haciendo click en el botón identificado con 'mine'.

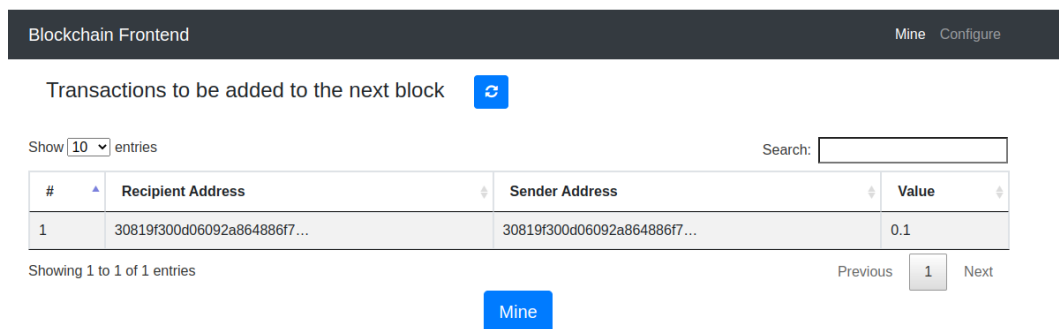


Imagen 10. Tabla de transacción por minar por el nodo

Una vez minado, el nodo recibe la recompensa o incentivo y se añade al bloque la transacción y se actualiza la blockchain y se propaga ésta a los demás nodos de la red para que estén sincronizados.

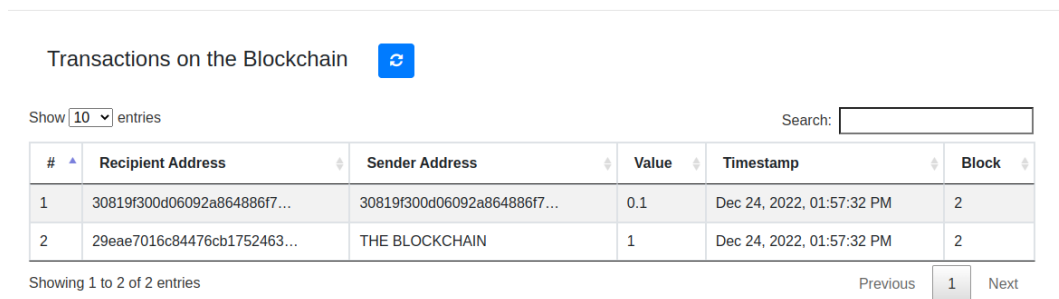


Imagen 11. Tabla de transacciones registradas en la blockchain del nodo

Cliente web - Interfaz de usuario

El proyecto cuenta con la estructura básica de una aplicación en angular y hace uso de las siguientes rutas:

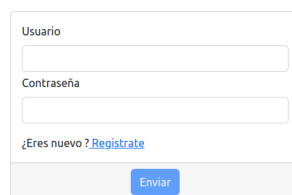
- /sign-in: Permite el inicio de sesión de usuarios en la plataforma
- /sign-up: Permite el registro de nuevos usuario
- /exchange: Permite el intercambio entre activos
- /tokens: Permite la compra de tokens dentro del sistema

- /transfer: Permite realizar transferencias entre usuarios del sistema
- /balance: Permite ver los saldos o el balance del usuario actual

Interacción con el sistema

Para poder usar el sistema, los nuevos usuarios deben de realizar un registro simple, para esto deben ubicar el enlace “Registrate”, hacer clic. Serán direccionados al formulario de registro el cual deberán de completar y posteriormente hacer clic en “Registrar”

Inicio de sesión



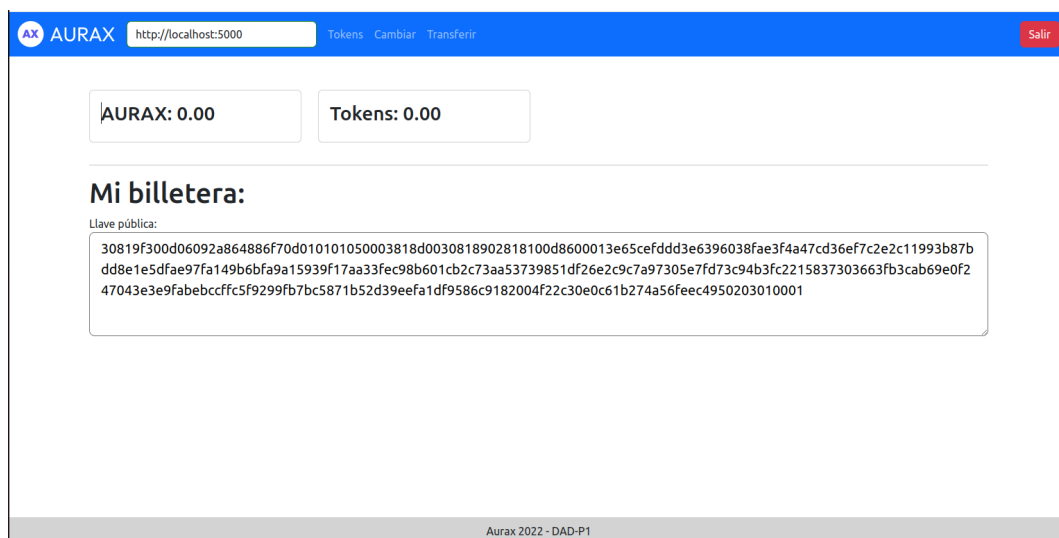
Formulario de inicio de sesión con los siguientes campos:

- Usuario:
- Contraseña:
- ¿Eres nuevo ? [Registrate](#)
- Botón:

Aurax 2022 - DAD-P1

Imagen 12. Formulario de registro

Una vez registrados iniciarán sesión inmediatamente y podrán ver el tablero de balance de saldos. En caso de usuarios previamente registrados podrán iniciar sesión a través del formulario de inicio de sesión.



Pantalla de balance de saldos con la siguiente información:

- Header: AX AURAX <http://localhost:5000> Tokens Cambiar Transferir
- Saldo: AURAX: 0.00 Tokens: 0.00
- Mi billetera:
 - Llave pública:
 - 30819f300d06092a864886f70d010101050003818d0030818902818100d8600013e65cefdd3e6396038fae3f4a47cd36ef7c2e2c11993b87b
 - dd8e1e5dfae97fa149b6bfa9a15939f17aa33fec98b601cb2c73aa53739851df26e2c9c7a97305e7fd73c94b3fc2215837303663fb3cab69e0f2
 - 47043e3e9fabebccffcf5f9299fb7bc5871b52d39eeffa1df9586c9182004f22c30e0c61b274a56feec4950203010001

Aurax 2022 - DAD-P1

Imagen 13. Pantalla de balance de saldos

Podremos identificar en el menú de navegación izquierdo 4 entradas identificadas como

- Tokens: Nos permitirá realizar el proceso de autogestión de tokens (compra de tokens con dólares)
- Transferir: Permite realizar transferencias de AURAX entre los usuarios del sistema
- Cambiar: Permite realizar el intercambio de activos (Tokens, AURAX y dólares)
- Salir: Permite cerrar sesión en el sistema.

Lo primero que debe hacer un nuevo usuario es adquirir tokens, por tanto deberá de dirigirse a la opción tokens en el menú y hacer clic en ella.

El usuario deberá de completar el formulario ingresando el monto en dólares que desea ingresar a la plataforma, así mismo el sistema calculará la cantidad de tokens a percibir de acuerdo al monto en dólares ingresado

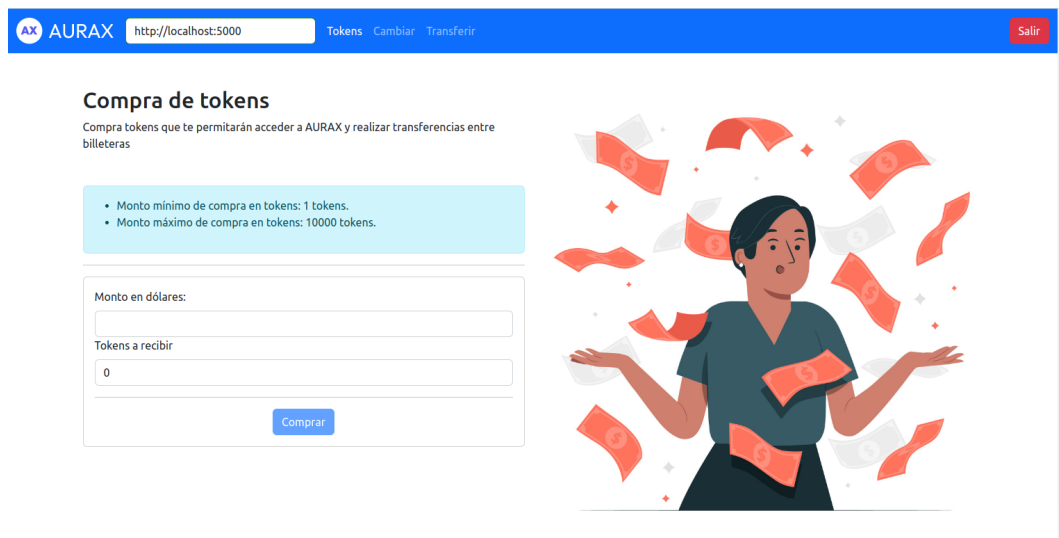
The image is a screenshot of a web browser displaying the AURAX application. The browser's address bar shows 'http://localhost:5000'. The application's header is blue with the 'AURAX' logo on the left and navigation links 'Tokens', 'Cambiar', and 'Transferir' in the center. A red 'Salir' button is on the right. The main content area is titled 'Compra de tokens' with a subtitle 'Compra tokens que te permitirán acceder a AURAX y realizar transferencias entre billeteras'. Below this, a light blue box contains two bullet points: 'Monto mínimo de compra en tokens: 1 tokens.' and 'Monto máximo de compra en tokens: 10000 tokens.' The form has two input fields: 'Monto en dólares:' and 'Tokens a recibir'. The 'Tokens a recibir' field currently shows '0'. A blue 'Comprar' button is at the bottom of the form. To the right of the form is an illustration of a person with dark hair and a blue shirt, surrounded by floating red and white banknotes.

Imagen 14. Formulario de compra de tokens

Realizada la compra de tokens el usuario ha de realizar un proceso de intercambio de activos. Puede realizar el intercambio de tokens a AURAX para poder realizar transferencias a otros usuarios a través de la blockchain o en caso de querer retirar sus activos podrá hacerlo realizando el cambio a dólares.

El sistema siempre indicará la cantidad a percibir luego de hacer la conversión a la moneda deseada.

Imagen 15. Formulario intercambio de activos

Si el usuario cuenta con una cantidad en AURAX podrá realizar transacciones con los otros usuarios registrados en la plataforma. Por facilidad y para efectos del proyecto los usuarios registrados son listados en un desplegable, sin embargo internamente el valor que se utilizará para realizar transacciones es la llave pública de la billetera que el usuario seleccionado tiene registrado.

Imagen 16. Formulario de transferencias

Conclusión

El proyecto realizado abarcó conceptos básicos de la tecnología blockchain así como de su uso centrado en el ámbito de las criptomonedas. Como se comentó desde un inicio, el proyecto estaba enfocado desde un aspecto académico y principalmente que sirva como una introducción a los procesos, tecnologías y conceptos usados en proyectos de criptomonedas.

La tecnología blockchain da pie a la posibilidad para distribuir y descentralizar aplicaciones permitiendo innovar sobre los mercados existentes o crear nuevos mercados. Siendo una de las tecnologías más nombradas de los últimos años se considera que es importante entender las ventajas que provee ante otras tecnologías y las consideraciones a nivel técnico que implican los desarrollos de esta índole.

Referencias

A.Moujahid, Marzo 2018 A Practical Introduction to Blockchain with Python
<http://adilmoujahid.com/posts/2018/03/intro-blockchain-bitcoin-python/>