UNIVERSIDAD NACIONAL
AUTÓNOMA DE MÉXICO

Facultad de Ingeniería

BLOCKCHAIN APPLICATION PROJECT

Blockchain-based singleplayer game

Alumnos:
Basile Álvarez Andrés José
Keller Ascencio Rodolfo Andrés

Profesora: Dra. Rocío Alejandra Aldeco Pérez

Grupo: 02

15 de junio de 2023

Índice

1. Introducción

A lo largo de nuestra clase de teoría de la materia de criptografía aprendimos que "*Blockchain* es una cadena de registros distribuidos y descentralizados unidos unos con otros que incluyen la firma digital de su creador. Una vez que estos registros son guardados no pueden ser alterados, ni en contenido ni en orden. Esto hace a *blockchain* un registro inmutable y de sólo agregación."Aldeco (2023a)

Blockchain cuenta con un gran número de propiedades, entre las cuales destacamos la confianza, verificación pública, descentralización, transparencia, integridad, redundancia y el anonimato. Dependiendo de las necesidades de una aplicación, se podría hacer uso de la *blockchain* debido a que cuenta con ventajas como el hecho de que al ser un tipo de tecnología descentralizada no hay un punto central de fallo; la confianza se basa en evidencia digital y no en terceras partes de confianza; es posible tener anonimato; y nos permite desarrollar aplicaciones a través de contratos inteligentes.

1.1. Objetivos

A lo largo de este proyecto se buscaba que los alumnos propusieran un prototipo de aplicación basado en *blockchain*, el cual fuera capaz de beneficiar a una organización, a la sociedad o a nuestra comunidad.

En este sentido, la propuesta realizada por los alumnos debía buscar el poder beneficiarse de las características y ventajas con las que cuenta la *blockchain*, para que, de esta manera, se mejorara un proceso que se realice de forma cotidiana.

1.2. Motivación

La propuesta que decidimos trabajar a lo largo de este proyecto se trata de un juego *single-player* en *blockchain*, el cual se basa en los juegos "*pick-a-door*", donde el jugador deberá de realizar una apuesta para que, posteriormente, proceda a elegir una de las tres puertas mostradas en pantalla y en caso de que el jugador resulte ganador, éste recibirá una recompensa donde inclusive su apuesta se verá multiplicada; en caso de resultar perdedor, el usuario deberá pagar la apuesta perdida.

La razón por la cual consideramos que el hecho de implementar esta aplicación en *blockchain* resultaría útil se basa en las propiedades con las que cuenta la *blockchain*, debido a que al ser una tecnología en la cual la información es pública, se cuenta con cierta confianza de que al estar jugando dentro de esta aplicación, se buscará tener una partida limpia y justa, siendo que los resultados y las apuestas realizadas pueden ser verificadas públicamente; al tratarse de una aplicación en *blockchain*, el usuario puede jugar de forma anónima desde cualquier parte del mundo con la única restricción de que deberá contar con fondos suficientes para realizar las apuestas. Esto, a su vez, le da cierto grado de seguridad debido a que puede acceder a la aplicación desde la comodidad de su casa, trabajo, o cualquier parte del mundo, por lo cual el monto de las apuestas que realice no lo pondrá en peligro y su dinero se encontrará seguro dentro de la plataforma. Por otro lado, la otra entidad, en este caso nosotros, contamos con la seguridad de que no se requiere nuestra presencia física al momento de que los jugadores hagan uso de la aplicación, por lo cual no se cuenta con un riesgo de asalto o algún tipo de ataque por parte de otros individuos.

Al hacer uso de esta tecnología, se evita el costo de la infraestructura que tendría, por ejemplo, un casino, el costo de los empleados, instalaciones, seguridad, entre otros. Se cuenta con el anonimato total por parte de los participantes, se cuenta con validación pública, se puede apostar cualquier monto

deseado y se tiene la certeza de estar participando en un juego justo dentro de una partida limpia, donde la probabilidad de ganar será siempre la misma, siendo estos algunos de los beneficios que se tienen al hacer uso de nuestra aplicación.

A través de las siguientes secciones hablaremos más detalladamente con respecto a la implementación de nuestra propuesta.

2. Metodología

2.1. Descripción técnica del prototipo

Para la realización de nuestra aplicación decidimos trabajar haciendo uso del protocolo Algorand, donde este protocolo define una red de nodos que implementan el *software* y protocolo Algorand. Se cuenta con dos instancias de esta red, *MainNet* y *TestNet*. Para efectos de nuestro proyecto decidimos trabajar dentro de la *TestNet*, la cual es utilizada para realizar pruebas contando con “Algos” falsos que pueden generarse sin la necesidad de comprarlas con dinero real. Un Algo se refiere a la moneda nativa de Algorand. Aldeco (2023b)

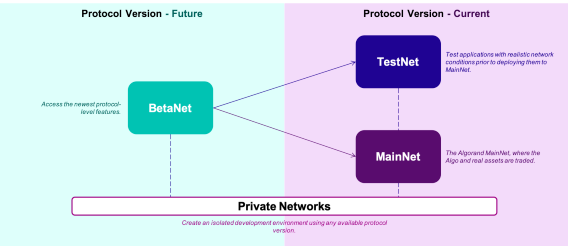

Figura 1: Tipos de redes en Algorand.

Algorand soporta oficialmente cuatro Kits de Desarrollo de Software (SDK), de los cuales nosotros hemos decidido trabajar con *Python*. De misma manera, para acceder a la cadena de bloques hicimos uso del servicio gratuito *PureStack API*.

El protocolo de Algorand se encuentra construido sobre tres conceptos fundamentales, entre los cuales encontramos las transacciones, los bloques y el consenso.

Las transacciones son la unidad básica de una cuenta en la red de Algorand, las cuales son utilizadas para transferir valores, siendo estas transacciones verificadas por los nodos participantes en la red. CoinTelegraph (2022)

Los bloques son grupos de transacciones que pasan a formar una única unidad, siendo éstos verificados por medio del algoritmo de consenso. El consenso es el algoritmo responsable de la verificación de los bloques de transacciones, el cual se asegura de que se cumplan con los requisitos del protocolo Algorand, este algoritmo beneficia a los usuarios que forman parte de su operación. CoinTelegraph (2022)

Algorand trabaja bajo el enfoque "*Pure proof-of-stake*", donde la influencia que tiene el usuario en la elección de un nuevo bloque es proporcional al número de *tokens* que posea el usuario en el sistema, siendo esto considerado como su participación. Cada usuario tiene la posibilidad de ser seleccionado, siendo su peso de propuesta y su voto proporcional a su grado de participación en la red. CoinTelegraph (2022)

A comparación del enfoque "*proof-of-work*", "*Pure proof-of-stake*" requiere una mucho menor cantidad de energía y poder de cómputo para la creación de bloques, haciendo que el proceso sea más sencillo, rápido y eficiente, siendo una de las razones por las cuales decidimos trabajar con este protocolo.

En el "*Pure proof-of-stake*", los usuarios son seleccionados de manera aleatoria y secreta con la idea de que logren proponer y votar bloques de transacciones. De esta forma, la seguridad de la red se

encuentra estrechamente ligada con la honestidad de los usuarios, donde si la mayor parte de los recursos de la red se encuentra en usuarios honestos, la red y el sistema permanecerá seguro. CoinTelegraph (2022)

Otras de las razones por las cuales decidimos hacer uso del protocolo Algorand fue debido a que se basaba en este enfoque, haciendo que los usuarios con menor grado de participación en la red no puedan dañar el sistema y los usuarios con mayor participación no deseen hacerlo debido a que podrían devaluar sus propios activos haciéndolo un protocolo relativamente seguro. Por otra parte, otra de las razones por las cuales decidimos utilizarlo fue por su facilidad de uso y las bibliotecas con las que se cuentan en *Python*.

Algorand puede ser utilizado para diferentes fines, entre ellos encontramos su uso para puntos de venta, procesos de facturación, gestión e intercambio de activos, pagos de IoT, subscripciones, firma de documentos, entre otros. A continuación se muestra un diagrama de dichos procesos y la manera en la cual se relacionan las aplicaciones con las redes, los usuarios y otros servicios.

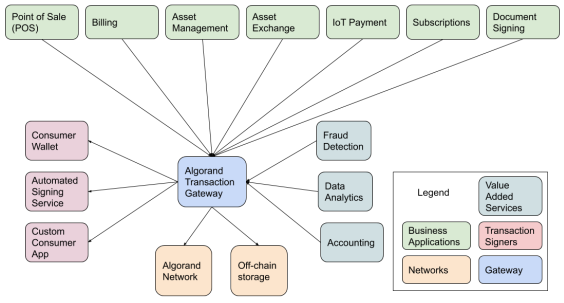

Figura 2: Organización y relación de elementos en Algorand. Gieseke (2020b)

Pasando ahora a una explicación más técnica de nuestra aplicación trabajada sobre *blockchain*, tendremos diferentes elementos como lo son los usuarios, las transacciones, los bloques, entre otros. Toda persona que quiera formar parte de la red de Algorand puede participar como usuario, así como puede participar en la elección, revisión y firma de bloques, como se mencionó anteriormente. Cada jugador de nuestra aplicación, al momento de perder o ganar, va a generar una nueva transacción, la cual al ser anexada a un bloque formará parte de los registros de la red. Existen siete tipos de transacciones, de los cuales nosotros vamos a utilizar la transacción para realizar pagos.

Deberá confirmarse la transacción y posteriormente el bloque para que el pago resulte exitoso y pueda anexarse a los registros previos. En este sentido, primero se inicia un proceso de pago y se crea la transacción que se desea realizar, ésta se envía a varios nodos dentro de la red y cada uno de estos nodos será el encargado de validar la transacción, después de haber sido validada por un nodo, este nodo lo enviará a otros nodos y será añadida a una cola de transacciones también conocido como *mempool*, siendo un área temproal de almacenamiento de transacciones antes de ser enviadas e incluidas en el siguiente bloque de la *blockchain*.

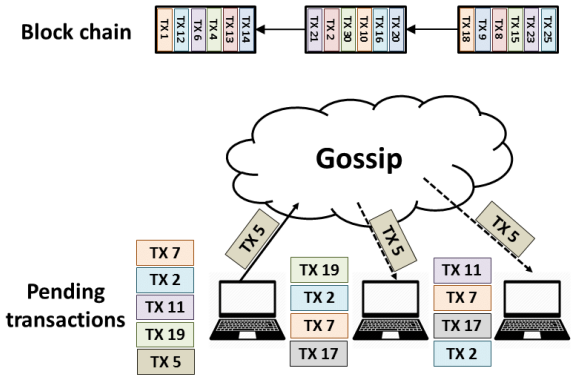

Figura 3: Transacciones y creación de bloques en Algorand. Hemo (2017)

El proceso de selección se lleva a cabo a partir de factores como la prioridad y la tarifa o cuota

por transacción, donde a partir del enfoque "*Pure proof-of-stake*", se elegirá a un usuario encargado de proponer un nuevo bloque para ser incluido en la *blockchain*, el cual será un bloque que se validará y se agregará a los registros de transacciones confirmadas por la red, siendo eliminada de la *mempool* para así terminar todo el proceso de pago o transacción.

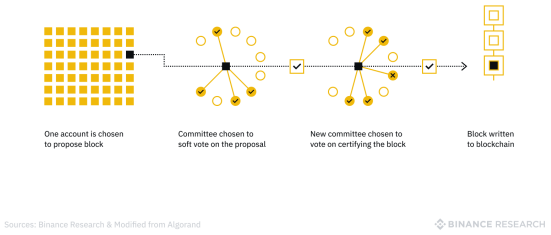

Figura 4: Votación de bloques en Algorand. Binance (2019)

La manera en como trabajan las transacciones será explicada posteriormente en la sección 2.2, 3.3 y 3.4 del presente documento.

2.2. Descripción técnica de la funcionalidad

En esta sección se hablará acerca de la funcionalidad de nuestro proyecto con respecto a la programación, diseño y transacciones por partidas ganadas y perdidas.

Lo primero que se realizó fue una ventana que simulara una conexión al *Wallet* de nuestro usuario o jugador, donde él deberá conectarse al programa mediante el uso de su par de llaves público-privada. Esta ventana únicamente simula el proceso que se realizaría al implementar una *Wallet* real, pero carece de la seguridad apropiada que implementan las *Wallets*. Dicha ventana cuenta con un buscador que permite seleccionar un archivo que contenga el par de llaves del usuario.

Tras haber cargado su par de llaves público-privada, se dirige al usuario a la ventana principal del juego, en la cual se cuenta con imágenes de tres puertas, imágenes de tres montos de apuesta, así como la información relacionada a la apuesta actual y el balance del jugador. El diseño de la implementación podrá apreciarse en la sección de resultados, mientras que en esta sección únicamente nos centraremos en la funcionalidad más importante, siendo estas las transacciones que se llevan a cabo y la obtención de los valores de balance del jugador.

2.2.1. Transacción partida ganada

Para las partidas ganadas se creó una función *client_wins(amount)*, que recibe como único parámetro el valor de la cantidad de Algos que ganó el usuario. Se va a hacer uso de la tasa mínima de cuota por transacción siendo esta de 1000 microAlgos, al haber ganado la partida, el *receiver* se refiere al jugador, por lo cual se toma la llave pública del jugador para poder realizar la transferencia, donde el monto a transferir será 1, 2 ó 3 Algos referente a la apuesta realizada por el usuario, multiplicado por dos, y mediante el uso de la función *transaction.PaymentTxn()*, pasaremos como parámetros la cuenta del casino que realizará la transacción y que pagará la cuota por transacción; los parámetros de la transacción, en este caso, el valor de la cuota que será de 1000 microAlgos; la llave pública del *receiver*, en este caso el jugador; y la cantidad de Algos a tranferir. Esta transferencia posteriormente se firmará mediante el uso de la función *sign()*, pasando la llave privada del casino y posteriormente mediante la función *wait_for_confirmation()*, se esperará la confirmación de que la transacción ha sido exitosa. Para todo esto, primero se tuvo que verificar que las cuentas tuvieran saldo suficiente, mediante el uso de la función *account_info_my_address['amount']*.

```
def client_wins(amount):
    global base
    params = algod_client.suggested_params()
    params.flat_fee = True
    params.fee = constants.MIN_TXN_FEE
    receiver = my_address # Players Account to receive
    note = "Client Wins".encode()
    amount = base * amount * 2
    if account_info_basile_keller.get('amount') < amount:
        print("Not enough funds from basile_keller_address, please wait!")
    unsigned_txn = transaction.PaymentTxn(basile_keller_address, params, receiver, amount, None,
    ↪  note)

    # sign transaction
    signed_txn = unsigned_txn.sign(private_key_basile_keller)

    #submit transaction
    txid = algod_client.send_transaction(signed_txn)
    print("Successfully sent transaction with txID: {}".format(txid))

    # wait for confirmation
    try:
        confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
    except Exception as err:
        print(err)

    print("Transaction information: {}".format(json.dumps(confirmed_txn, indent=4)))
    print("Decoded note:
    ↪  {}".format(base64.b64decode(confirmed_txn["txn"]["txn"]["note"]).decode()))
    print("Starting Account balance: {} microAlgos".format(account_info_my_address.get('amount')))
    print("Amount transfered: {} microAlgos".format(amount))
    print("Fee: {} microAlgos".format(params.fee))
```

2.2.2. Transacción partida perdida

Para las partidas perdidas se creó una función *client_looses(amount)*, que recibe como único parámetro el valor de la cantidad de Algos que perdió el usuario. Se va a hacer uso de la tasa mínima de cuota por transacción siendo esta de 1000 microAlgos, al haber perdido la partida, el *receiver* se refiere al casino, por lo cual se toma la llave pública del casino para poder realizar la transferencia, donde el monto a transferir será 1, 2 ó 3 Algos, dependiendo del tipo de apuesta que haya realizado el jugador, y mediante el uso de la función *transaction.PaymentTxn()*, pasaremos como parámetros la cuenta del jugador que realizará la transacción y que pagará la cuota por transacción; los parámetros de la transacción, en este caso, el valor de la cuota que será de 1000 microAlgos; la llave pública del *receiver*, en este caso el casino; y la cantidad de Algos a tranferir. Esta transferencia posteriormente se firmará mediante el uso de la función *sign()*, pasando la llave privada del jugador y posteriormente mediante la función *wait_for_confirmation()*, se esperará la confirmación de que la transacción ha sido exitosa. Para todo esto, primero se tuvo que verificar que las cuentas tuvieran saldo suficiente, mediante el uso de la función *account_info_my_address['amount']*.

```
def client_looses(amount):
    global base
    params = algod_client.suggested_params()
    params.flat_fee = True
    params.fee = constants.MIN_TXN_FEE
    receiver = basile_keller_address # Basile Keller Casino´s Account to receive
    note = "Client Looses".encode()
    amount = base * amount

    if check_balance(amount):
        unsigned_txn = transaction.PaymentTxn(my_address, params, receiver, amount, None, note)

        # sign transaction
        signed_txn = unsigned_txn.sign(private_key_my_address)

        #submit transaction
        txid = algod_client.send_transaction(signed_txn)
        print("Successfully sent transaction with txID: {}".format(txid))

        # wait for confirmation
        try:
            confirmed_txn = transaction.wait_for_confirmation(algod_client, txid, 4)
        except Exception as err:
            print(err)

        print("Transaction information: {}".format(json.dumps(confirmed_txn, indent=4)))
        print("Decoded note:
         ↪  {}".format(base64.b64decode(confirmed_txn["txn"]["txn"]["note"]).decode()))
        print("Starting Account balance: {}
         ↪  microAlgos".format(account_info_my_address.get('amount')))
        print("Amount transfered: {} microAlgos".format(amount))
        print("Fee: {} microAlgos".format(params.fee))
```

3. Resultados

En esta sección se presentan capturas de pantalla del resultado final de la implementación de nuestra idea. Se ha dividido en subsecciones para poder apreciar con mayor detalle cada uno de los procesos que se realizan en el proyecto.

3.1. *Login y Wallet*

Partiendo de la descripción de la funcionalidad de nuestra aplicación, inicialmente tuvimos que crear una simulación de uso de *Wallet* de manera que el usuario contara con una ventana de *login* en la cual él fuera capaz de cargar un archivo que contara tanto con su llave pública como con su llave privada. Al tratarse de una aplicación que requiere realizar transacciones por parte tanto del usuario como de la entidad dueña del juego, se requiere ambos par de llaves para realizar las transacciones.

De esta forma, se diseñó el siguiente simulador de *Wallet* para la carga del archivo del usuario.

Figura 5: Ventana de *Login* de nuestra aplicación. Simulación de *Wallet*.

Es muy importante que el usuario tenga la seguridad de que la información relacionada con su llave privada de la cuenta de Algorand va a seguir siendo privada y confidencial, es por esto por lo cual se implementa una *Wallet*, siendo la razón por la cual nosotros hemos realizado este simulador en el cual el usuario realiza la carga de su información mediante un archivo.

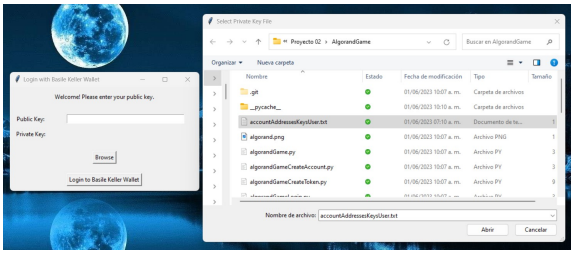

Figura 6: Selección de archivo con llave pública y privada de usuario.

Tras el proceso de ingreso de datos mediante la carga de archivos, el usuario puede verificar la llave pública que se está cargando a la plataforma y de esta manera comenzar con su proceso de apuesta.

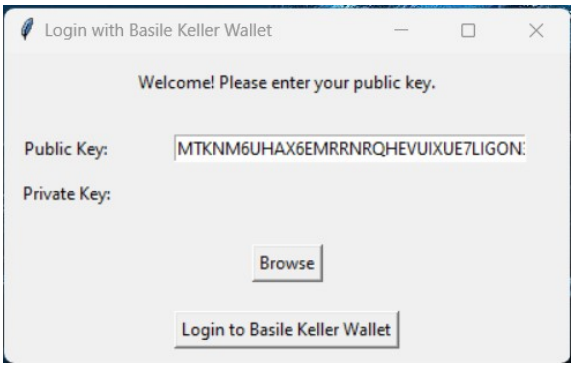

Figura 7: Lectura y confirmación de llave pública de usuario.

Si las credenciales ingresadas por el usuario son correctas, podrá entrar a nuestra aplicación. Se verifica si el usuario cuenta con los fondos necesarios para poder participar en el juego, y en caso de que así suceda, el usuario podrá comenzar a jugar.

3.2. Inicio de Partida

Tras haber ingresado correctamente, al usuario le aparece una nueva pestaña en la cual podrá apreciar la aplicación que hemos creado, en este caso el juego *single-player* basado en juegos de tipo "*Pick-a-Door*". Dentro de esta interfaz se cuentan con tres puertas, tres monedas representando el grado de la apuesta, el valor de la apuesta y el balance del jugador.

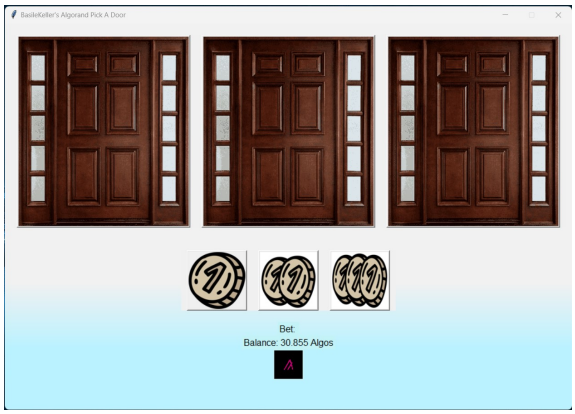

Figura 8: Pantalla inicial de juego

Lo primero que deberá realizar el usuario será seleccionar el grado de apuesta que desea realizar. Se cuenta con tres tipos de apuesta, la apuesta 1, la apuesta 2, y la apuesta 3. En este sentido, el usuario apostará 1, 2 ó 3 Algos, respectivamente. Para esto, el sistema verifica si el usuario cuenta con los fondos necesarios para poder realizar la apuesta.

Tras la elección del monto de la apuesta, el usuario procederá a la elección de la puerta deseada.

3.3. Partida Ganada

En caso de que el usuario haya seleccionado una puerta ganadora, se mostrará un mensaje en el cual se felicitará al usuario y se le hará saber que ha ganado. En caso de ganar, su apuesta inicial se verá duplicada.

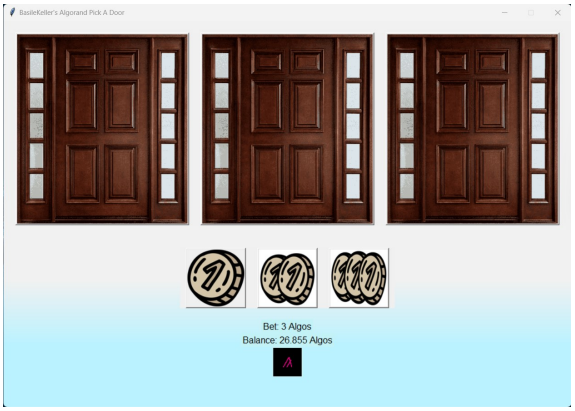

Figura 9: Seleccionando un valor de apuesta.

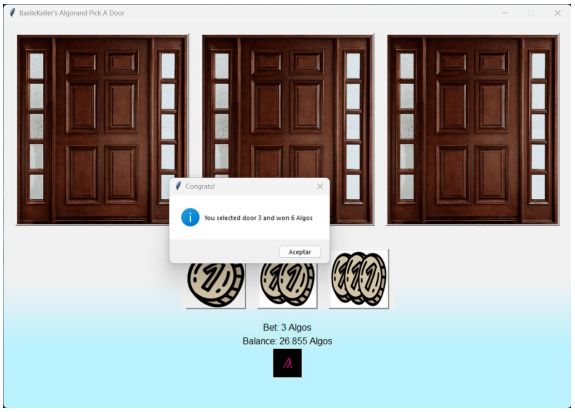

Figura 10: Ganando partida

Al cerrar el mensaje de felicitación, se observa la actualización del balance del usuario.

Simultáneamente, dentro de la terminal se aprecia el proceso de transacción, en el cual se indica el identificador de la transacción. Dentro de la información de la transacción encontramos un valor de ronda de confirmación; el monto de microAlgos que se transfieren; la tarifa o cuota por la realización de la transacción (cubierta por el casino, pues es la entidad que perdió y deberá realizar el pago); los valores de ronda entre las cuales la transacción resultará válida, siendo estas las rondas 30521745 y 30522745; la red sobre la cual nos encontramos realizando las transacciones, siendo esta *TestNet*; los

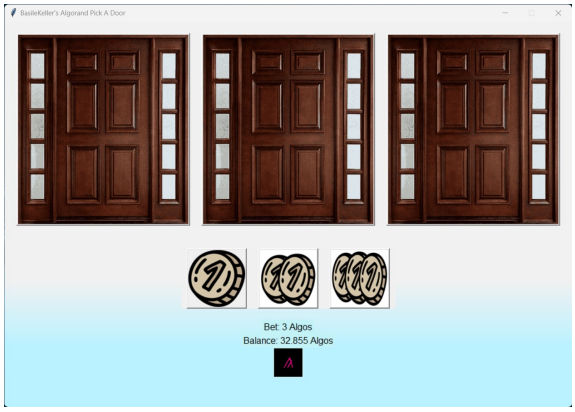

Figura 11: Actualización de balance por partida ganada

valores de las llaves públicas que mandan y reciben los fondos de la transferencia; y el tipo de transferencia que en este caso sería un pago. Gieseke (2020a)

Por otro lado, nosotros hemos decidido poner notas más claras que indican que el cliente ha ganado, el balance inicial de la cuenta del cliente al momento en que ganó, la cantidad de microAlgos transferidos y la tarifa o cuota de la transferencia.

Figura 12: Transacción por partida ganada

3.4. Partida Perdida

En caso de que el usuario haya seleccionado una puerta perdedora, se mostrará un mensaje en el cual se indicará al usuario que no ha resultado ganador. En caso de perder, el casino obtendrá el valor de la apuesta inicial del usuario.

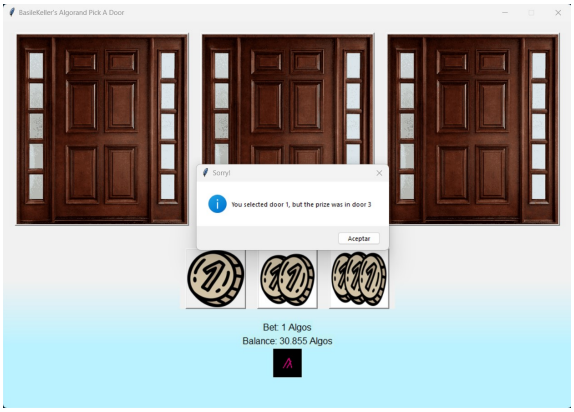

Figura 13: Perdiendo partida

Al cerrar el mensaje en el cual se le indica que ha perdido, se observa la actualización del balance del usuario.

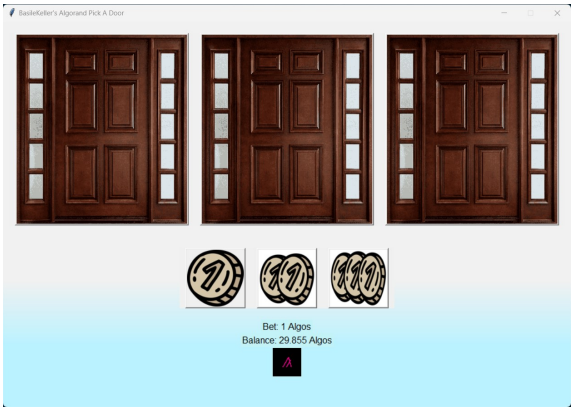

Figura 14: Actualización de balance por partida perdida

Simultáneamente, dentro de la terminal se aprecia el proceso de la transacción, en el cual se indica el identificador de la transacción. Como habíamos visto en la sección anterior, dentro de la información de la transacción encontramos un valor de ronda de confirmación; el monto de microAlgos que se transfieren; la tarifa o cuota por la realización de la transacción (cubierta por el jugador, pues es la entidad que perdió y deberá realizar el pago); los valores de ronda entre las cuales la transacción resultará válida, siendo estas las rondas 30521695 y

30522695; la red sobre la cual nos encontramos realizando las transacciones, siendo esta *TestNet*; los valores de las llaves públicas que mandan y reciben los fondos de la transferencia; y el tipo de transferencia que en este caso sería un pago. Gieseke (2020a)

De misma manera, en notas más claras indicamos que el cliente ha perdido, el balance inicial de la cuenta del cliente al momento en que perdió, la cantidad de microAlgos transferidos y la tarifa o cuota de la transferencia a realizar.

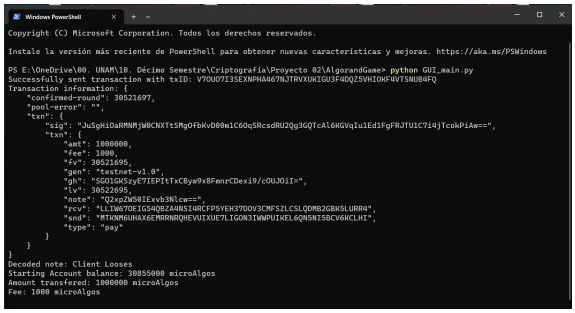

Figura 15: Transacción por partida perdida

4. Conclusiones y recomendaciones finales

A través de este proyecto se buscó desarrollar una aplicación basada en *blockchain*, la cual mejorara algún proceso que se realice de manera cotidiana a partir de las propiedades y ventajas que nos ofrece esta tecnología.

Como equipo, decidimos implementar un juego *single-player* basado en la idea de los juegos "*Pick-a-Door*", donde buscábamos aprovechar las propiedades de la *blockchain* para obtener beneficios a partir de la idea de descentralización, verificación pública, confianza, transparencia, integridad, redundancia y anonimato para poder obtener una aplicación que reemplace los juegos de apuesta presenciales, por ejemplo, en casinos, promoviendo la seguridad, así como un juego justo y limpio, donde al tratarse de un juego realizando en *blockchain* se disminuye una gran cantidad de gastos en infraestructura, instalaciones, traslados, salarios, entre otros.

Para esto, se utilizó la *blockchain* de Algorand, donde para trabajar nuestra propuesta se decidió realizar un prototipo en la red *TestNet*, evitando así el pago por "Algos" para la prueba de nuestra aplicación. De misma forma, el proyecto fue desarrollado en el lenguaje de programación *Python*, aprovechando las bibliotecas y funciones preexistentes.

Tras la realización de nuestro trabajo, consideramos que la aplicación presenta excelentes bases, pues es una aplicación que busca beneficiar de múltiples formas a los usuarios, tanto clientes como casinos, pues ambos pueden obtener ganancias a partir de una actividad que busca fomentar el juego limpio y justo, verificable públicamente pero que mantenga el anonimato de ambas partes, siendo además una manera de aportar seguridad al proceso y evitar la presencia de ambas partes en un mismo lugar físico, lo cual también permite reducir los costos de infraestructura, salarios, equipos, entre otros.

En un futuro, la aplicación podría implementar una *Wallet* real (el usuario no debería de ingresar su llave privada en la aplicación, sino que debería de haber una integración con una *Wallet* para que el juego pueda conocer la cuenta del usuario y que éste pueda pagar en caso de perder), y podría hacer uso de la red *MainNet*, para obtener beneficios reales, sin embargo, se debería continuar con los procesos de prueba y evaluación por parte de usuarios. De misma manera, para futuras versiones se podría mejorar la interfaz del juego e inclusive crear una interfaz que permita una mayor cantidad de juegos y aplicaciones para el uso y disfrute de los usuarios.

Finalmente, consideramos que este proyecto nos permitió conocer a mayor profundidad lo que es una *blockchain* y el tipo de acciones, actividades, aplicaciones o transacciones que se pueden realizar en ella, así como el proceso mediante el cual se genera una transacción, la forma en cómo las transacciones son verificadas, anexadas a bloques y cómo los bloques de transacciones son elegidas para posteriormente incorporarse a los registros de la *blockchain*. Este fue un proyecto muy completo debido a que nos abrió el panorama y permitió que nos adentráramos al mundo de las *blockchain* a partir de herramientas conocidas como lo es *Python*.

Referencias

Aldeco, D. R. (2023a). 5.4 fundamentos de blockchain. universidad nacional autónoma de méxico - facultad de ingeniería.

Aldeco, D. R. (2023b). Algorandesp. - https://github.com/raldecop/algorandesp.git - universidad nacional autónoma de méxico - facultad de ingeniería.

Binance (2019). Algorand (algo) - https://research.binance.com/en/projects/algorand - binance research.

CoinTelegraph (2022). What is the algorand blockchain, and how does it work? - https://cointelegraph.com/news/what-is-the-algorand-blockchain-and-how-does-it-work - cointelegraph.

Gieseke, E. (2020a). Algorand structure - https://developer.algorand.org/docs/get-details/transactions/ - developer portal.

Gieseke, E. (2020b). Creating a point-of-sale application with the algorand blockchain - https://developer.algorand.org/solutions/creating-point-sale-application-algorand-blockchain/ - developer portal.

Hemo, R. (2017). Transaction flow in algorand - https://www.researchgate.net/figure/an-overview-of-transaction-flow-in-algorand$_{f}ig1_{3}20362651 - researchgate$.