UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Prototipo para crear y automatizar procesos de verificación de credenciales en *blockchain*

Trabajo de graduación presentado por Augusto Estuardo Alonso Ascencio para optar al grado académico de Licenciado en en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2022

UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Prototipo para crear y automatizar procesos de verificación de credenciales en *blockchain*

Trabajo de graduación presentado por Augusto Estuardo Alonso Ascencio para optar al grado académico de Licenciado en Ingeniería en Ciencia de la Computación y Tecnologías de la Información

Guatemala,

2022

Vo.Bo.:

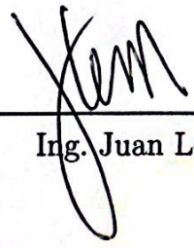
(f) _____

Ing. Roberto Chiroy

Tribunal Examinador:

(f) _____

Ing. Roberto Chiroy



(f) _____

Ing. Juan Lemus

(f) _____

Ing. Douglas Barrios

Fecha de aprobación: Guatemala, 8 de diciembre de 2022.

Índice

Lista de figuras v

Lista de cuadros vi

Resumen vii

Abstract viii

1. Introducción 1

2. Justificación 2

3. Objetivos 4
- 3.1. Objetivo general . . . 4
- 3.2. Objetivos específicos . . . 4

4. Alcance 5

5. Marco teórico 6

6. Metodología 20
- 6.1. Diseño . . . 20
- 6.2. Desarrollo . . . 23
  - 6.2.1. Desarrollo con el diseño inicial . . . 23
  - 6.2.2. Cambio a SDK en JS de Metaplex . . . 23
  - 6.2.3. Desarrollo utilizando el SDK en JS de Metaplex . . . 24
  - 6.2.4. Desarrollo de *dashboard* para los usuarios . . . 25
- 6.3. Implementación . . . 28
  - 6.3.1. Aplicaciones web . . . 28
  - 6.3.2. Api . . . 28

7. Resultados 29
- 7.1. Costos . . . 29
  - 7.1.1. Costos arquitectura prototipo . . . 29
  - 7.1.2. Costos arquitectura actual . . . 32
- 7.2. Velocidad . . . 33
  - 7.2.1. Velocidad de prototipo . . . 33
  - 7.2.2. Velocidad de emisión credenciales arquitectura actual . . . 33

7.2.3. Velocidad de validación de credenciales . . . 33
7.3. Escalabilidad . . . 34
7.3.1. Pruebas de estrés . . . 34
7.4. Encuestas . . . 34
7.4.1. Utilidad del SDK . . . 34
7.4.2. Encuestas a trabajadores del gobierno . . . 37
7.5. Prototipos . . . 37

8. Conclusiones **38**

9. Recomendaciones **39**

10. Bibliografía **40**

11. Glosario **41**

Lista de figuras

1. Actividades que dependen de una verificación de identidad . . . 2

2. Estructura básica de Blockchain . . . 8
3. Estructura básica de Blockchain con su índice . . . 8
4. Diferencia de resultados del algoritmo sha256 . . . 9
5. Cadena de blockchain con *hashing* . . . 9
6. Cifrado con la llave privada . . . 12
7. Información que se guarda en un *Account* . . . 15
8. Estructura de almacenamiento de tokens de *Solana's Token program* . . . 16
9. PDA . . . 17
10. Estructura de almacenamiento de tokens de *Solana's Token program* una cuenta para su metadata . . . 18
11. Visualización de un NFT y sus cuentas . . . 18

12. Proceso de usuario interactuando con un dashboard . . . 21
13. Ejemplo proceso usuario creando/editando un NFT . . . 22
14. Flujo para validación de fecha de expiración . . . 26
15. Proceso final de usuario interactuando solicitando una credencial de usuario . . . 27

16. ¿Ha escuchado de solana? . . . 34
17. ¿Consideraría útil un SDK para el desarrollo? . . . 35
18. ¿Desarrolla en Javascript? . . . 35
19. ¿Considera fácil de utilizar el SDK "i-am-verifiable-button"? . . . 36
20. ¿Considera que para utilizar el SDK"i-am-verifiable-button" . . . 36
21. Resultados de encuestas a trabajadores . . . 37

Lista de cuadros

1. Desglose backend 29
2. Desglose costos dashboard interno 30
3. Desglose costos dashboard para usuarios 30
4. Costo arquitectura prototipo 31
5. Costo operadores registrales 32
6. Mantenimiento oficinas registrales 32
7. Tiempos actividades en prototipo 33
8. Tiempos actividades en RENAP 33
9. Validaciones otorgadas por el RENAP 33
10. Tiempos con 10 solicitudes simultaneas 34
11. Tiempos con 100 solicitudes simultaneas 34

Resumen

Un sistema de verificaciones eficiente, seguro y descentralizado significaría un nuevo paso a cómo operaríamos en nuestro día a día; en un mundo en el que dependemos mucho en poder verificar quiénes somos desde cuándo ingresamos a un hospital hasta cuando deseamos hacer una compra necesitamos verificar nuestra identidad. Con el ingreso de web 2.0 se ha buscado la manera de lograrlo de una manera eficiente, pero debido a su naturaleza se es imposible tenerlo descentralizado causando que no exista una única manera de verificar credenciales, además de que la centralización de los datos ha causado que los usuarios pierdan poder donde siempre hay algún regulador tomando las decisiones. *Blockchain* nos otorga parte de la solución para desarrollar un sistema que sea capaz de crear credenciales, emitirlas y verificarlas de una manera más descentralizada y sin necesidad de un regulador empoderando a los usuarios y dándoles mayor confianza en los procesos de verificación. Principalmente logramos obtener los beneficios en el *blockchain* con los NFT que es un estándar en *blockchain* para simular tokens no fungibles. Los NFT han revolucionado los últimos años el mundo de los bienes digitales, a pesar que se ha generado mucha polémica debido a los precios que muchos de estos han alcanzado. Los NFT han demostrado su funcionalidad en habilitar una manera de poder verificar su autenticidad. En este proyecto se propone hacer uso de solana como red de *blockchain* y el uso de los NFT para poder crear un prototipo de un sistema descentralizado de verificación de credenciales. Se creará un grupo de sistemas que permitan crear las credenciales con un estándar definido para poderlas diferenciar del resto de NFTS sin complicar su desarrollo también emitir las credenciales para los usuarios y verificar las mismas de una manera rápida, segura y descentralizada.

Abstract

An efficient, secure, and descentralized verification system would mean a new step in how we live our day-to-day tasks in a world that requires verification in many aspects of our life. From shopping for things online to getting into a hospital, we need to validate some credentials to prove who we are. With the start of Web 2.0, people have searched for a way to achieve this in an efficient way, but the limitations of this technology mean that it can't be descentralized, meaning that each application will have its own data and logic for their verification process. Web 2.0 also requires a regulator in many of its operations. Blockchain gives us part of the solution to this problem by developing a system that is capable of creating, giving, and validating credentials in a decentralized way without the need for a regulator in between the processes, giving more power to the users. As a result of this, they also have more confidence in each process. However, we primarily benefit from blockchain by using NFTs, which is a standard for creating non-fungible tokens that are typically associated with an asset. Despite the controversy caused by the high prices some of them have received, NFTs have revolutionized all types of digital assets over the last few years. The NFT's have demostrated their funcionality in prividnig a way to verify the autenticity of these assets. That's why in this project it is proposed to use Solana blockchain and their nfts to create a prototype of a descentralized system of credentials verification. A set of systems will be created to create credentials with a defined standard in order to differentiate them from the rest of the NFTS without overcomplicating them too much, as well as to allow users to mint these credentials and verify them in a fast, secure, and decentralized manner.

CAPÍTULO 1

Introducción

Los procesos de verificación de credenciales son los que nos permiten que a través de un proceso la persona que esté detrás del mismo sea realmente quien menciona ser y no caer en un fraude de identidad. Dicho proceso es necesario para muchos de nuestros aspectos en la vida desde aspectos profesionales, económicos hasta más personales como viajes y recreación.

Este proceso se ha buscado automatizar desde que la irrupción del internet ocurrió, pero siempre las limitaciones de lo que nosotros conocemos como web 2.0 nos han impedido poder estandarizar y centralizar este proceso. En su mayoría de casos siempre se ha necesitado proseguir con participación de organizaciones y terceros por ejemplo una agencia los cuales colectan información de un usuario para poder determinar con base en dichos documentos si la identidad es real o no llegando a ser bastantes efectivos pero gastando mucho más recursos en este caso (Slomovic, 2014).

En este trabajo se buscará poder marcar los primeros pasos con el desarrollo de un prototipo, en el cual se especifique como se puede automatizar los procesos de verificación de identidad por medio del uso del blockchain específicamente la red de Solana que tiene un TPS de 2,036 (Solana, 2022b) asegurándonos velocidad para la escalabilidad del proyecto y aprovechar las propiedades del blockchain como su inmutabilidad para poder tener menores costos de operaciones, mayor velocidad en trámites, descentralizar la información y tener transparencia en todas las transacciones que se lleven a cabo. Se buscará poder demostrar su funcionalidad con documentos como:

- Carne del IGGS
- DPI
- Licencia
- Antecedentes penales
- Pasaportes

Abriendo la posibilidad de automatizar los procesos de estos documentos y muchos más a través de *smart contracts*.

CAPÍTULO 2

Justificación

Según Shah et al., n.d en su paper *"Providing a Secure, Reliable and Decentralized Document Management Solution Using Blockchain by a Virtual Identity Card"* con los sistemas actuales de manejo de identidades hay varias industrias que sufren de estos modelos que no cumplen con nuestras necesidades siendo estas:

- **Gobierno**
- **Bancos**
- **Hospitales**
- **Educación**

Figura 1: Actividades que dependen de una verificación de identidad

También el formato actual para poder verificar VID's posee problemas de privacidad esto debido a que para poder llevar a cabo estos procesos se requiere la recolección de datos personales y eso resulta en preocupaciones sobre el manejo de la misma al poder darle un uso inapropiado al no ser completamente transparente los procesos de verificación y sus resultados y puede otorgar la entrega de información personal a terceros (IEEE, 2014, 2).

Blockchain es una tecnología basada en criptografía que esta enfocada en la descentralización de datos, seguridad, inmutabilidad. Dicha tecnología se popularizo con la creación del Bitcoin en el 2009 y desde entonces ha estado en constante evolución.

Solana es un blockchain lanzado en marzo del 2020 que busca proveer un ecosistema de desarrollo eficiente, escalable y económico a comparación de las alternativas. Algunos de los beneficios que el blockchain de Solana nos otorga para combatir estos problemas a la hora de tener VID's en el mismo:

- Descentralización de información.
- VID universal
- Acceso autorizado donde solo se muestra lo que se desea
- No hay dependencia de un ecosistema físico. No importa si se pierden documentos físicos.
- Menos corrupción y fraude
- Reducción de costos.
- Velocidad de transacción.
- Más confianza. La descentralización de los datos resulta en que no haya un regulador de por medio que tome decisiones unilaterales en como realizar el proceso de verificación.
- Inmutabilidad

CAPÍTULO 3

Objetivos

3.1. Objetivo general

Con este proyecto se tiene como meta poder proveer un prototipo funcional de un sistema de manejo de VID bajo blockchain a pesar de que el mismo presenta ciertos retos y se ha visto como una opción no viable debido a su escalabilidad

3.2. Objetivos específicos

- Demostrar que los NFT pueden ser utilizados para verificar la identidad de un usuario por medio de sus credenciales (licencia, dpi, pasaporte).
- Tener una aplicación prototipo web que conforme con el protocolo de los NFT para verificar las credenciales de un usuario.
- Demostrar que la metadata de un NFT puede ser utilizada para validar fechas de expiración y requisitos.
- Desarrollar un SDK que facilite la verificación de una credencial para los desarrolladores.

CAPÍTULO 4

Alcance

En este proyecto se muestra como se puede utilizar los NFT para verificar credenciales. Estas credenciales se crean a través de una aplicación web dirigida para entidades emisoras en las cuales pueden configurar su nombre, precio, descripción, expiración y requisitos que estos necesiten. Dentro de la configuración no se encuentra habilitada la opción de vincular más documentos a los NFT lo que podría ser útil para documentos que necesiten el respaldo de otros. La publicación de las credenciales se generan para acceso público lo cual es útil para le emisiones de credenciales como pasaporte, DPI, etc.

La emisión de los documentos se encuentra habilita a través de otro *dashboard* en el cual cualquier usuario puede acceder y si cumple con los requisitos se le permite emitir el documento deseado.

Estas se alimentan de un api hecho en ExpressJS la cual nos permite tener nuestra llave privada segura.

Por último, se concluye el proyecto con una librería de JS y VueJS para habilitar la verificación de credenciales emitidas por nuestra web app. Esta se acompaña de una *landing page* para facilitar su uso.

El prototipo esta habilitado para funcionar en la devnet de Solana quitando cualquier valor existente en las transacciones realizadas.

CAPÍTULO 5

Marco teórico

Desde la creación y el auge de la web 2.0 en 2000, la cual permite: interoperabilidad entre X y Y, facilidad de acceso y familiaridad de uso para que el usuario participe con el contenido creado desde una página web; se ha trabajado para lograr una implementación que permita usar identidades virtuales. Páginas como Microsoft, Amazon, Netflix, Facebook han implementado sus propias identidades virtuales ya sea para poder interactuar con otros usuarios como es el caso de Facebook o para manejar servidores en la nube. Estas aplicaciones poseen las limitaciones de la tecnología de web 2.0 las cuales son:

- Saturación de contenido.
- Bases de datos relacionales.
- Las plataformas se encuentran gobernadas por autoridades centralizadas.
- Falta de privacidad. Por ejemplo el uso de *cookies* en nuestro navegador permite rastrear a los usuarios.

Las bases de datos relacionales se encuentran gobernadas por las autoridades respectivas de las aplicaciones implicando que la propiedad de información se encuentre centralizada y no tengamos control en como se maneja esa información. Esta centralización en conjunto con la falta de consenso en lo que una identidad virtual debe de ser genera que cada aplicación implemente un modelo de identidad virtual que puede diferir al resto para una sola persona complicando el proceso de verificación para el usuario.

La relevancia de poder verificar la autenticidad de una identidad virtual de una manera segura, rápida y eficaz ha crecido ante la globalización en el mundo y el impacto que estas aplicaciones originadas en el boom del web 2.0 ha ocasionado. La Figura 1 nos muestra diferentes categorías de actividades que necesitan la verificación de identidad que van desde poder tener acceso a servicios de salud hasta el alquilar un vehículo en un carro (C. Marlene Fiol, 2005).

La variedad de actividades en las que tenemos que otorgar información para poder verificar una identidad en las aplicaciones web 2.0 nos da una idea de cuanta información nuestra queda comprometida y puede ser consultada sin nuestro conocimiento.

Según Ashforth y Mael (1989) la identificación de una persona es su sentido de pertenencia a una categoría social. Nosotros como individuos utilizamos estas categorías sociales para describirnos

a nosotros mismos en términos de similitudes con miembros de nuestro grupo en contraste con otras categorías sociales.

La coherencia en la creación de una identidad es importante en un ambiente virtual debido a la reducción de contacto físico entre miembros (C. Marlene Fiol, 2005). Basado en los *frameworks* teóricos que se tienen disponibles para poder modelar una identidad hay 4 niveles distintos de identidad; individual, relacional, social e identidad material (C. Marlene Fiol, 2005).

- **Individual:** hace referencia a las características, metas, rasgos de cada individuo.
- **Relacional:** se refiere a los roles en los que nos manifestamos a través de interacciones sociales.
- **Social:** es el concepto que se tiene a "auto otorgar" determinado por la percepción de varios grupos sociales, incluyendo la nacionalidad, etnia, religión, familia. Esta nos permite mantener una sensación de pertenencia y nos motivemos a actuar de cierta manera.
- **Material:** como nosotros asociamos aspectos materiales como carros, ropa, propiedades a nuestra identidad.

Las identidades virtuales juegan un papel importante nuestra identidad ya que realmente las fronteras físicas son eliminadas a través de nuestros perfiles, avatares, nuestra identidad virtual podemos moldearlas y mostrarlas al mundo como nosotros deseemos bajo nuestro deseo sin limitaciones físicas (C. Marlene Fiol, 2005).

Un sistema de manejo de identidades principalmente debe de ser seguro, confiable y estandarizado. Estas características se encuentran ausentes en los sistemas actúales que las aplicaciones web 2.0 nos ofrece.

Según Shah et al. (2020) es posible modelar un sistema que cumpla con los requisitos mencionados utilizando blockchain. El eslabón más debil en el sistema actual de identidades se encuentra en las instituciones gubernamentales como los bancos y agencias de crédito. En estas instituciones el involucramiento de terceros sin autorización del usuario es lo acostumbrado, con la información privada siendo guardada en una ubicación desconocida y vulnerable a ser hackeada.

Según Shah et al. (2020) las industrias que sufren en los sistemas actuales de verificación de identidad son los siguientes:

- Gobierno: El tiempo y costo requerido para poder procesar una verificación incrementa al involucrar diferentes niveles del gobierno en la forma de burocracia.
- Bancos: Esta se convierte menos segura ya que requiere detalles como el login para poder verificar lo cual puede ser algo completamente hackeable. Este tipo de verificación el cual es el más popular se ha visto comprometido por:
  - Malas prácticas en contraseñas.
  - Falta de caracteres especiales.
  - Mal manejo de las contraseñas en las bases de datos como falta de cifrado.
  - Hackeo de los sistemas que tienen acceso a las contraseñas.
- Hospitales: Los centros de salud de alta calidad no son accesibles a todas las personas. La interoperabilidad entre el personal del hospital, el espacio para verificación y otros procesos retrasa el tratamiento y causa ineficiencia.
- Educación: Los certificados de educación se encuentran cada vez más vulnerables ante falsificaciones. Según Grolleau et al. (2008) en estados unidos se producen más de 2 millones de falsificaciones de diplomas universitarios en los Estados Unidos.

Blockchain se podría describir como una lista de registros de información que esta en continuo crecimiento. Dicha lista se encuentra conformada de bloques que son los que contienen dichos registros de información que se encuentran organizados en un orden cronológico y están vinculados y asegurados entre sí por medio de pruebas criptográficas (Binance, 2022).

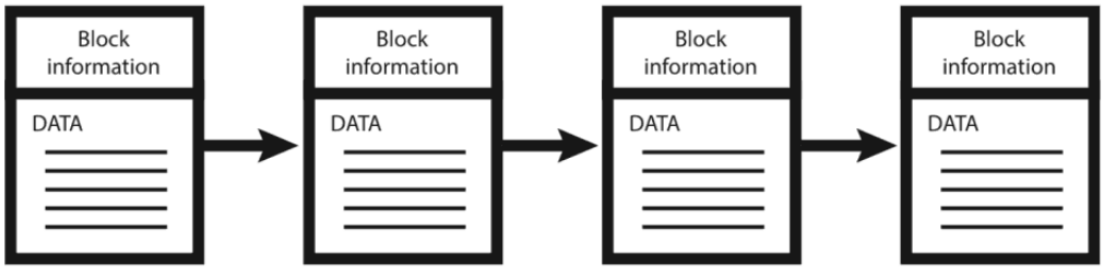


Figura 2: Estructura básica de Blockchain

(Summers, 2022)

Los bloques de blockchain se encuentran guardados como archivos estos se encuentran indexados por un archivo que se encuentra fuera del blockchain. Por ejemplo Bitcoin posee un archivo llamado blkindex.dat el cual posee el índice para los archivos del blockchain. La información de los bloques se encuentra guardados en el archivo blk000n.dat, donde 000n representa el número de bloque. Esto es importante ya que nos permite la propiedad de inmutabilidad, al separar el vínculo del siguiente bloque y en su lugar guardar la información en un archivo por separado no tenemos que alterar el bloque pasado (Summers, 2022).

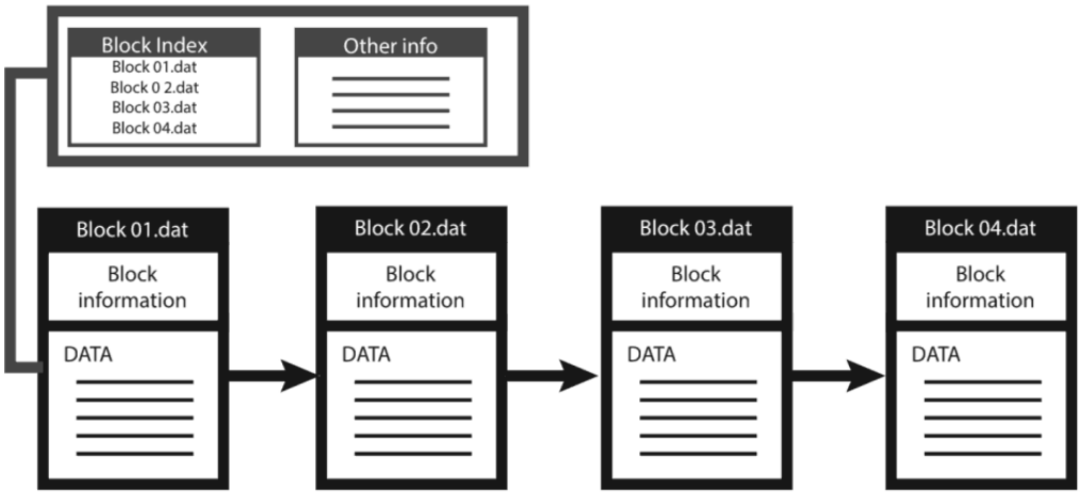


Figura 3: Estructura básica de Blockchain con su índice

(Summers, 2022)

La criptografía sirve como base en blockchain para su seguridad, integridad e inmutabilidad. En nuestro día a día no es necesario conocer todas las aplicaciones que la criptografía tiene dentro del blockchain, únicamente las más importantes. Uno de los aportes de la criptografía más destacados es el *hashing*. Según Summers (2022) *hashing* es el nombre que se le otorga a un proceso en el que

a partir de un determinado valor de ingreso obtenemos un valor de retorno que posee siempre un tamaño fijo de caracteres sin importar el input ingresado. Esto lo logra a partir de un algoritmo en el que lo tomaremos un poco como caja negra y se enfocará en explicar los principios que logran su funcionamiento.

El valor de retorno de un algoritmo de *hashing* es un número grande pero usualmente se representa por un valor alfanumérico, volviéndolo más fácil de leer y utilizar. Otro principio es que si se ingresa el mismo valor siempre se obtendrá el mismo resultado sin embargo si en el valor de ingreso hay un cambio por mas mínimo que sea el resultado será completamente diferente 4 (Summers, 2022).

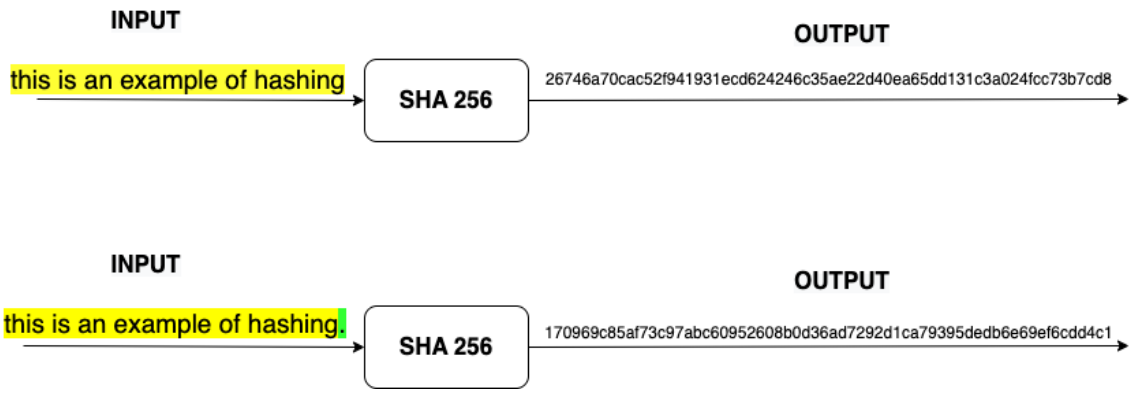


Figura 4: Diferencia de resultados del algoritmo sha256

Otro aspecto importante del *hashing* es que el proceso de generar un valor de retorno a partir de un valor de ingreso se logra fácilmente, en opuesto a el proceso de a partir de cierto valor de retorno obtener el valor de ingreso es difícil de calcular si no es que imposible con el poder computacional que se posee hoy en día. Debido a esto el *hashing* se puede referir a un identificador digital único para la información y utilizarse para verificar su integridad (Summers, 2022).

Se puede pensar que ya que la cantidad de información que existe estos algoritmos pueden dar el mismo resultado para diferentes valores de ingreso lo cual es cierto, cuando esto sucede se le llama *collisions* y este depende en el algoritmo de *hashing* que se este utilizando. En blockchain se utiliza algoritmos de *hashing* como sha256 con 2^256 posibilidades (Summers, 2022).

La principal función del *Hashing* dentro del blockchain es asegurar la integridad de la información después de que haya sido escrita dentro del blockchain. Si describimos su implementación de la manera más simple dentro de nuestra lista de bloques cada uno tiene un *hash* correspondiendo al bloque pasado a excepción del primero que posee una cadena de ceros, a dicho bloque se le conoce como el bloque génesis (Summers, 2022).

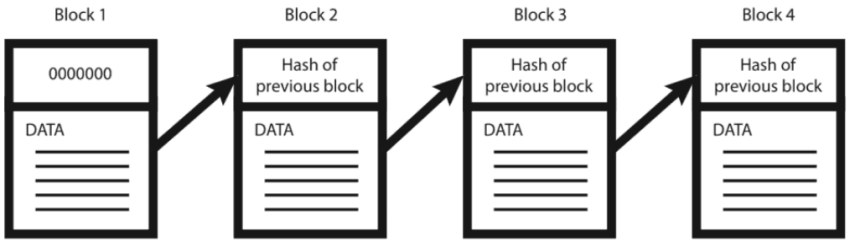


Figura 5: Cadena de blockchain con *hashing*

(Summers, 2022)

Si observamos la Figura 5 si alguien intentara cambiar el contenido del bloque número dos el *hash* iría a cambiar y sería incorrecto para el resto de bloques.

Otro de los usos más destacados de la criptografía en el blockchain son la firmas digitales. Para comprenderlas es necesario primero hablar acerca de el algoritmo de Rivest–Shamir–Adleman (RSA) el cual la mayoría de redes de blockchain utiliza como base en sus algoritmos.

En 1978 Ron Rivest, Adi Shamir y Leonard Adleman introducen un algoritmo criptográfico en el que esencialmente se buscaba remplazar el algoritmo *National Bureau of Standards (NBS)* que brindaba menos seguridad. RSA implementa un criptosistema de llave pública así como firmas digitales (Milanov, 2009).

Introducida en un tiempo donde la era de los correos electrónicos se esperaban tomar mayor relevancia RSA implementaba dos ideas importantes:

1. Cifrado de llave pública. Esta idea omite la necesidad de un "*courier*" para poder entregar las llaves a los recipientes sobre un canal seguro antes de trasmitir el mensaje original. En RSA las llaves de cifrado son publicas, mientras que las llaves de descifrado no lo son, de tal manera solo la persona que posea la llave correcta podrá descifrar el mensaje (Milanov, 2009). Todos deben de poseer su pareja de llaves de cifrado y descifrado. Dichas llaves deben de ser realizadas con la premisa en que la llave privada no se pueda deducir a partir de la llave pública de manera sencilla.

2. Firmas digitales. El destinatario de un correo puede necesitar verificar que un mensaje transmitido se origino de la persona que se espera. Esto se logra utilizando la llave privada del remitente, luego la firma puede ser verificada por cualquiera usando la llave pública del remitente (Milanov, 2009).

Según Milanov (2009) RSA el par de llaves con las que se logran los procesos de cifrado y descifrado son un set de números especiales. Estos los entenderemos como E y D, donde E corresponde a nuestra llave pública y D a nuestra llave privada. Dentro de el proceso empezamos con nuestro mensaje que lo simbolizaremos como M, es el que se busca "cifrar". Hay cuatro procesos que son específicos y esenciales a un criptosistema de llave pública:

a. Descifrar un mensaje cifrado debe de darte el mensaje original, específicamente.

$$D(E(M)) = M \qquad \text{(ecuación 1)}$$

b. Revertir el procedimiento produce de igual manera M:

$$E(D(M)) = M \qquad \text{(ecuación2)}$$

c. E y D son fáciles de calcular.

d. El hecho de que E sea público no compromete el secretismo de D, significando que no es fácil de averiguar D a partir de E.

A partir de una una E dada aun no hemos obtenido una manera eficiente de calcular D. Si C = E(M) es un texto cifrado, entonces tratar de encontrar D por medio de satisfacer M en E(M) = C tiene una dificultad no razonable: el número de mensajes a probar sería impráctico. Una E que logre satisfaga los puntos (a), (c) y (d) se le llama "*trap-door one-way function*" y también es una "*trap-door one-way permutation*". Esto debido a que la inversa de D es fácil de calcular si cierta información secreta ("*trapdoor*") se encuentra disponible, pero de otra manera es difícil. Este es de una vía ya que es fácil de calcular en una dirección pero difícil en la dirección opuesta. Es una

permutación porque satisface (b), significando que cada texto cifrado es un mensaje potencial, y cada mensaje es un cifrado de otro mensaje (Milanov, 2009).

Para que el algoritmo de RSA logre una firma digital debe de asegurarse que el mensaje se originó a partir de determinado remitente y no fue enviado a través de el a partir de un tercero que utilizo la llave pública del recipiente.

Si bob desea mandar a Alice un mensaje privado. Para Bob firmar el documento *rsa* se descifra un mensaje con la llave privada, esto permitido por las propiedades (a) y (b), que nos asegura que cada mensajes es un texto cifrado de otro mensaje y que cada texto cifrado puede ser interpretado como un mensaje.

Interpretando los siguientes símbolos:

- D_B: Llave privada de Bob.
- E_B: Llave pública de Bob.
- D_A: Llave privada de Alice.
- E_A: Llave pública de Alice.
- S: Firma digital

$$D_B(M) = S. \qquad \text{(ecuación 3)}$$

Luego ciframos S con la llave pública de Alice:

$$E_A(S) = E_A(D_B(M)). \qquad \text{(ecuación 4)}$$

Esta manera nos aseguramos que solo Alice puede descifrar el documento. Cuando lo realice ella. obtendrá la firma generada por medio de $D_A((E_A(D_B)(M))) = S$. De esta manera Alice sabe que el mensaje vino de Bob, ya que únicamente su llave privada podría haber generado la firma. El mensaje no necesita ser enviado de manera separada, debido a que Alice puede deducir por medio de la firma utilizando la llave pública de Bob, formalmente $E_B(S) = E_B(D_B(M)) = M$. Debido a que S depende de M, y la transmisión cifrado que Bob envió depende de S, se tiene una transmisión que depende de ambos el mensaje y la firma, así que ambos pueden ser deducidos del documento transmitido (Milanov, 2009).

Esto nos asegura que el mensaje no pudo ser modificado, ya que un M modificado en la forma de M' también hubiera generado una firma $S' = D_B(M')$, la cual es imposible ya que Alice no conoce D_B por la propiedad (d). De tal manera que no solo Alice posee prueba de que Bob firmo el mensaje y el fue el remitente, si no que también no puede modificar M o generar una firma para cualquier otro mensaje (Milanov, 2009).

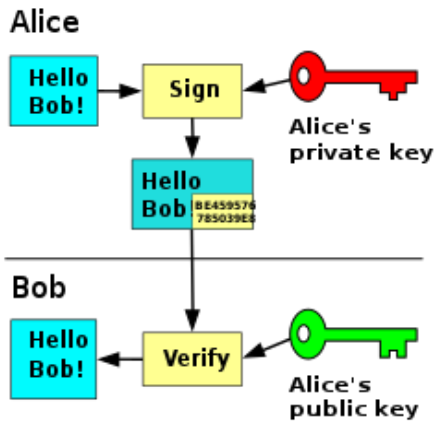


Figura 6: Cifrado con la llave privada

Como se puede observar Figura 6 Alice en este ejemplo cifra su mensaje con su llave privada y Bob que es el recipiente es capaz de verificar la autenticidad de la fuente por medio de la llave pública de Alice que le ha compartido.

Para comprender la seguridad que el algoritmo otorga es necesario hablar de la matemática que la respalda.

A partir de ahora debemos representar M como un valor numérico para poder operar operaciones aritméticas en el. Ahora representemos M como un número entero entre 0 y n - 1. Sean e, d, n números positivos. La llave pública E es la pareja (e, n) y la llave privada D es (d, b), n = pq (San Jose State University, 2022).

Ahora debemos cifrar el mensaje por medio de elevándolo a la potencia de e y aplicándole modulo de n para obtener C que es nuestro mensaje cifrado y luego descifrar C elevando C a la potencia de d modulo n para obtener M de nuevo.

$$C = E(M) = M^e \ mod\, n.$$
$$M = D(C) = C^d \ mod\, n. \qquad \text{(ecuación 5)}$$

Podemos observar que estamos preservando el mismo tamaño de información, ya que M y C son números enteros entre 0 y n-1, por congruencia modular. Las parejas de llaves (e,n) y (d,n) son diferentes para cada usuario pero acá solo s considerará el escenario general.

Para poder crear la llave de cifrado primero elegimos dos números de manera "aleatoria" p y q, multiplicamos y producimos n = pq. A pesar que n es público, este no revelara p y q ya que prácticamente es imposible factorizar a partir de n asegurándonos así que d es prácticamente imposible de derivar de e.

Ahora para obtener los valores apropiados para e y d. Escogemos d ser un número entero de gran valor, que debe de ser un coprimo a (p - 1) * (q - 1), significando que la siguiente ecuación debe de ser satisfecha.

$$gcd(d, (p-1) * (q-1)) = 1. \qquad \text{(ecuación 6)}$$

El término "gcd" lo conocemos como el divisor más grande en común.

Se buscará calcular e a partir de d, p y q, donde e es un inverso multiplicativo de d. Esto quiere decir que necesita satisfacer.

$$e * d = 1 (mod\, \phi(n)). \qquad \text{(ecuación 7)}$$

Acá introducimos la función de totiente de Euler $\phi(n)$, donde el valor de retorno indica la cantidad de números positivos menores a n que son coprimos con n. Para los números primos p, este resultado es $\phi(p) = p - 1$. Para n, nosotros obtenemos de la función que:

$$\begin{aligned} \phi(n) &= \phi(p) * \phi(q) \\ &= (p-1) * (q-1) \\ &= n - (p+q) + 1. \end{aligned} \qquad \text{(ecuación 8)}$$

De esta ecuación podemos sustituir $\phi(n)$ dentro de la ecuación número 7 y obtener.

$$e * d = 1 (mod\, \phi(n))$$

lo cual es equivalente con

$$e * d = k * \phi(n) + 1$$

para determinado número entero k.

Por los teoremas de aritmética modular, el multiplicativo inverso de a modulo m existe solo y solo si a y m son coprimos. Por lo tanto como d y $\phi(n)$ son coprimos d posee un multiplicativo inverso en el anillo de enteros modulo $\phi(n)$.

Por ahora ya hemos comprobado que

$$\begin{aligned} D(E(M)) &= (E(M))^d = (M^e)^d \ (mod\, n) = M^e d \ (mod\, n) \\ E(D(M)) &= (D(M))^e = (M^d)^e \ (mod\, n) = M^e d \ (mod\, n) \end{aligned}$$

También ya que $e * d = k * \phi(n) + 1$ se puede sustituir en las ecuaciones para obtener

$$M^{\mathrm{ed}} = M^k \phi(n) + 1 \ (mod\, n)$$

Claramente nosotros queremos mostrar igualdad a M. Para probar esto necesitaremos una identidad debido a euler y Fermat: para cada número entero M coprimo a n se obtiene

$$M^{\phi}(n) = 1 \ (mod\, n). \qquad \text{(ecuación 9)}$$

Debido a que nosotros especificamos previamente que M debe de ser mayor a 0 y menor a n, sabemos que M no sería coprimo de n si y solo si M fuera p o q, en los números enteros de ese intervalo. Por lo tanto las probabilidades que M sea p o q están en el mismo orden de la magnitud a 2/n. Esto significa que M es prácticamente definitivo un primo relativo a n, por lo tanto la ecuación 9 se mantiene y utilizándolo evaluamos:

$$M^{ed} = M^{k\phi(n)+1} = (M^{\phi(n)})^k M = 1^k M \ (mod\, n) = M$$

Resulta que esto funciona para todas las M. Por lo tanto E y D son permutaciones inversas y nos asegura que el secreto de D no se ve comprometido a partir de exponer E.

Con esto mencionado podemos asegurar que las firmas digitales dentro de blockchain nos ayudan a asegurar nuestra identidad en la red, otorgándonos una manera de probar si la información realmente proviene de nosotros sin comprometer nuestra llave privada.

Otro concepto importante del blockchain para poder entender como funciona es la Tecnología de Contabilidad Distribuida ya que esta nos permite tener descentralizada la información. Se hablará brevemente de este tema ya que solo necesitamos entender la idea principal en nuestro día a día. Estos son sistemas que permiten a programas y usuarios tener transacciones relacionado a archivos. Este guarda información en múltiples ubicaciones en cualquier punto del tiempo, eso quiere decir que a diferencia de las bases de datos tradicionales la información no se encuentra centralizada (101 Blockchains, 2021).

Esta descentralización de la información crea la necesidad de un consenso para poder decidir que información es válida y que información no lo es para agregar al blockchain. Las personas que tengan computadores en la red de blockchain son incentivados a verificar transacciones a cambio de una recompensa. Este proceso en los primeros protocolos de blockchain como Ethereum y Bitcoin se le conoce como *Proof of Work* donde todos los computadores en la red están activamente participando minando el *hash* correspondiente al nuevo bloque, este proceso sería fácil si únicamente se buscara un *hash* cualquiera y permitiría ataques para saturar la red (DDoS) es por eso que se agrega al mismo dificultad en el algoritmo permitiendo agregar un bloque a la red cada cierto tiempo (Yakovenko, 2022).

Uno de los grandes retos que tiene el blockchain es la velocidad en la que puede realizar sus transacciones esto debido a que este proceso de verificación es costoso e ineficiente. Los nuevos protocolos de blockchain han implementado otras soluciones como lo es *Proof of Stake*. A diferencia del algoritmo de *Proof of Work* los computadores únicamente actúan como validadores y en lugar de todos estar participando como competencia a quien encuentra resuelve primero el algoritmo se escoge un validador de una manera pseudo aleatoria. En este proceso no cualquier computador puede actuar como validador para poder participar es necesario dejar un *stake* dentro de la red, si un validador aprueba transacciones fraudulentas pierde su stake (Yakovenko, 2022).

Solana es una red que utiliza *Proof of Stake* en conjunto a *Proof of history* para transacciones más rápidas y eficientes. Solana tuvo la creación de su primer bloque en marzo 16 del 2020 entrando como directa competidora a la red de Ethereum con la meta de poder quitar las limitaciones que este nos presentaba al utilizar el consenso de Proof of Work.

Proof of history es un concepto importante en el diseño de Solana. Según Yakovenko (2022) es una secuencia de computación que puede proveer una manera de verificar el pasaje del tiempo a través de la criptografía entre dos eventos. Este utiliza una función criptográfica segura para que el valor de retorno no pueda ser predicho del valor de ingreso y debe de ser ejecutado en su totalidad para generar el output.

La función se corre en una secuencia en un solo core, y su previo valor es su entrada, periódicamente grabando el valor retornado actual y cuantas veces ha sido invocado. Este valor luego puede ser recalculado y verificado por computadoras externas en paralelo chequeando cada secuencia del segmento en un core por separado.

Los datos se les puede grabar una marca de tiempo dentro de esta secuencia por medio de agregando datos (o un *hash* de algunos datos) dentro del estado de la función. El registro del estado, su índice y datos de cuando fueron agregados dentro de las secuencias provee una marca de tiempo

que garantiza que que fue generado antes de que el siguiente *hash* de la secuencia en el blockchain fuera generado (Yakovenko, 2022).

Otro aspecto importante que hay que tener en cuenta en solana es la capacidad de memoria que la red de Solana posee teóricamente la capacidad de almacenar 10 billones de *Accounts* en 640GB de memoria. Debido a esto en la red de solana hay que pagar un fee por almacenar los datos que ingresemos a la red, a dicho monto se le conoce como la renta.

En Solana los registros que nos permiten guardar el estado entre transacciones se les llama *Accounts*. Los *Accounts* son muy similares a los archivos en un sistema operativo que pueden tanto como guardar información o pueden ser un programa de igual manera que un archivo los *Accounts* incluyen metadata que indica quienes tienen permitido acceder a dicha información y de que manera (Solana, 2022a).

Field	Description
`lamports`	The number of lamports owned by this account.
`owner`	The program owner of this account.
`executable`	Whether this account can process instructions.
`data`	The raw data byte array stored by this account.
`rent_epoch`	The next epoch that this account will owe rent.

Figura 7: Información que se guarda en un *Account*

Como se observa en la figura número 7 estos pueden guardar lamports como un banco, nos indican quién es el dueño de esta nueva *Account* si no se provee uno se otorga como dueño "*System Program*" que es un programa ya incluido en el código de Solana, un flag indicándonos si es un ejecutable o no y por último el epoch en el que se nos ira cobrando la renta por guardar esta información en la red de solana (Solana, 2022a).

Para poder continuar es necesario aclarar ciertos términos dentro del blockchain de Solana:

- **Nodo:** Un nodo en solana se le llama a cualquier computadora corriendo el software necesario para poder conectarse al cluster de solana.
- **Validador:** Se le denomina validadores a los nodos en solana que hayan aportado un stake para poder participar en el procesamiento de transacciones y del consenso de *Proof of Service*.
- **Solana Cluster:** Un cluster de solana es un conjunto de validadores trabajando en conjunto para poder servir transacciones del cliente y mantener la integridad de la red.

Para poder interactuar con la red de Solana se utiliza RPCs. Estos *requests* son manejados por nodos que usualmente se encuentran dedicados únicamente a la tarea de manejar estos *requests* y no participar en el consenso. Sin embargo esto es importante ya que un mal rendimiento de estos *requests* no es diferente a un mal rendimiento de un cluster en Solana. Es por eso que Solana nos

ofrece principalmente dos categorías de proveedores. Los proveedores públicos, estos son útiles para demos, pequeños programas y programas beta privados y los privados que a diferencia de los públicos estos no son gratis pero nos aseguran una mejor experiencia para el usuario y es útil para aplicaciones de gran escala (Solana, 2022c).

Otro concepto importante a saber son los *Smart Contracts*. Según IBM (2022) un smart contract se le denomina a un programa que esté guardado en el blockchain que corre cuando ciertas condiciones se cumplan. Usualmente se utilizan para automatizar la ejecución de un acuerdo en el que todos los participantes puedan estar seguros del resultado sin necesidad de un intermediario. La red de solana es capaz de soportar el desarrollo de smart contracts y que entre los mismos se pueda interactuar lo cual se discutirá después.

Hemos hablados bastante de como interactuar con la red de blockchain, que son los smart contracts, como funciona la red pero aún no hemos hablado de como podemos tener control de nuestro balance y nuestros bienes digitales en la red. Las billeteras de criptomonedas nos permiten esto además de poder recibir, gastar, enviar criptomonedas y guardar las llaves privadas de un usuario. Estas billeteras pueden ser digitales o físicas y utilizan conexión a internet para poder acceder la red de blockchain. Su utilización más popular son las billeteras digitales con ellas también podemos tener control de nuestros NFT, observarlos incluso enviarlos.

Para poder llevar a cabo la virtualización de las credenciales utilizando blockchain con la descripción anteriormente mencionada de las identidades virtuales es necesario encontrar un protocolo que cumpla con los requisitos para poder imitar este comportamiento; solana nos ofrece el estándar de los tokens fungibles y los tokens no fungible (Non fungible tokens) en solana mejor conocido como *Solana's Token program*. Dicho protocolo consiste en poder crear tokens no fungibles, cada aspecto de nuestra identidad nos pertenece solo a nosotros y no puede ser sustituido por el de alguien más. Dentro de blockchain el acto de publicar un NFT a la red se le conoce como *minting* para que pueda ser comprado, vendido o intercambiado.

La estructura para poder almacenar los tokens fungibles y no fungibles consiste en esencialmente 3 *Accounts* dentro de solana. El *Account* que representa a un usuario, *Mint Account* esta cuenta es la que guarda la información de determinado token es decir sus autoridades en la cuenta, cuantos tokens hay disponibles etc y el *Token Account* que es una cuenta vinculada a la billetera del usuario y a un determinado *Mint Account* esta cuenta guarda la información de cuantos tokens se es dueño esa billetera como podemos observar en la figura número 8 (Metaplex, 2022). El *Mint Account* recibe su nombre de la acción al publicar el NFT.

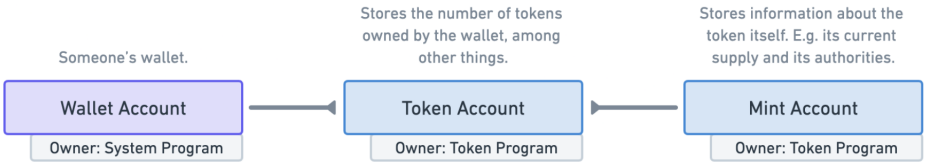


Figura 8: Estructura de almacenamiento de tokens de *Solana's Token program* (Metaplex, 2022)

Para que estos tokens nos sean útiles es necesario tener la capacidad de poder describirlos de una manera eficiente y segura. Metaplex nos provee un programa llamado *Token Metadata* para poder lograr agregar metadata a tokens fungibles o no fungibles. Este lo logra por medio de unos *Accounts* especiales llamados *Program Derived Accounts* PDA que en escénica le permiten a un programa firmar transacciones a nombre de ciertos *Accounts* sin necesidad de una llave privada (Metaplex, 2022). Además los PDA sirven como la fundación para permitir la invocación entre programas.

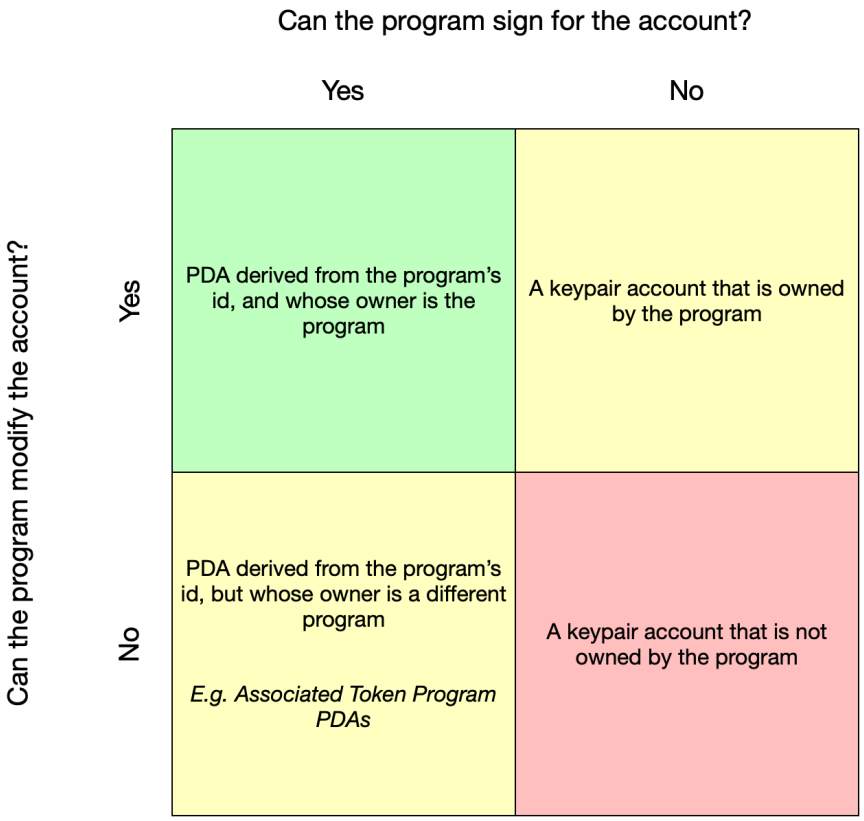


Figura 9: PDA

(Solana Cookbook, 2022)

Para entender el concepto de estos PDA es importante mencionar que más que creados son encontrados. Los Program Derived Account (PDA) se generan a partir de una combinación de semillas y el id de un programa. Esta combinación se pasa a travez de una función sha256 para poder ver si se genera una llave pública que se encuentre en la curva de la elíptica ed25519.

Este proceso tiene un 50 % de probabilidad de que se termine con una llave pública en la curva de la elíptica en este caso se vuelve a intentar generarlo por medio de agregar un poco de ruido a nuestro valor de ingreso hasta que se encuentre una llave pública que no se encuentre en la curva con el fin de poder tener una manera determinista de encontrar el mismo PDA (Solana Cookbook, 2022).

Como observamos en la Figura 8 aún no hemos definido varios aspectos como el nombre de determinado token, si es mutable o no, quienes son sus creadores etc.

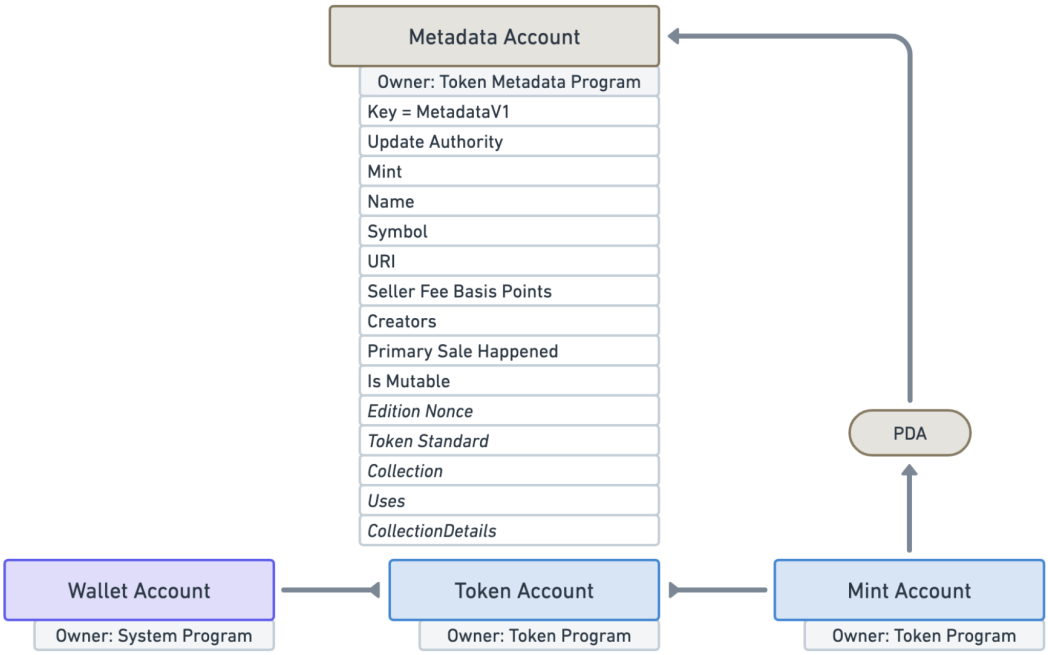


Figura 10: Estructura de almacenamiento de tokens de *Solana's Token program* una cuenta para su metadata

(Metaplex, 2022)

El programa de Metadata de Metaplex ya nos provee un *Account* que se adhiere al *Mint Account* por medio de un PDA Figura 10. Esta metadata nos guarda información importante para su interacción en el ecosistema de Solana; como su nombre, una lista de creadores que poseen un atributo de si el token ha sido firmado o no por dicho creador, a que colección pertenece entre otros.

Previamente hablamos de como el costo de almacenaje de información en el blockchain de solana tiene un costo (renta). Entre mayor sea el espacio ocupado mayor será la renta a pagar es por eso que en nuestro *Metadata Account* que nos guarda la metadata del NFT no podemos guardar información como una imagen, un archivo que se asocie a este nuevo token; para ello se cuenta con el campo «URI» que apunta a un archivo JSON fuera del blockchain permitiendo de esta manera guardar información adicional sin estar restringidos a los costos de lo que la renta de almacenamiento de dichos datos involucraría (Metaplex, 2022).

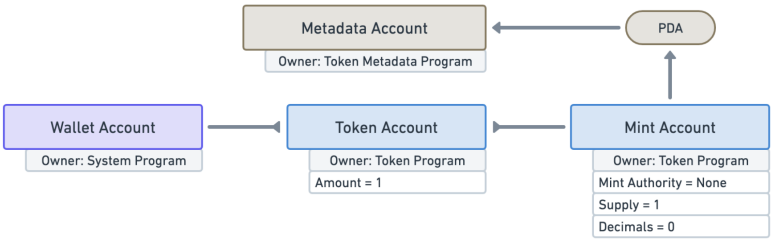


Figura 11: Visualización de un NFT y sus cuentas

(Metaplex, 2022)

Utilizando Metaplex logramos crear un NFT como se logra observar en la Figura 11 ya que cumple con las características de un NFT. Siendo estos un solo token disponible en el *Mint Account* con cero decimales y sin una autoridad en el *Mint Account* significando que nadie podrá generar tokens adicionales asegurándonos que sea único.

Dentro del proceso de la creación de un NFT el programa *Token Metadata* de Metaplex nos crea un tercer *Account* llamado *Master Edition Account* siendo este un PDA del *Mint Account* al igual que el *Metadata Account* y asigna como autoridad *Mint Account* asegurando que nadie pueda mintear ni congelar tokens sin pasar antes por el programa actuando como prueba de no fungible (Metaplex, 2022).

CAPÍTULO 6

Metodología

6.1. Diseño

El objetivo principal de este proyecto se tiene el poder tener un prototipo funcional para poder facilitar la emisión y automatización de procesos que requieren una verificación de identidad para poder demostrar la utilidad del blockchain en este caso de uso e incentivar a futuros estudiantes/profesionales a seguir desarrollando sobre el proyecto y lograr poder llevar a cabo a producción una versión de este proyecto.

Dicho modelo se espera realizar por medio de entidades que sean «emisoras» estas serán encargadas de crear los NFT (documentos, bienes vinculados en el blockchain, etc), estos bien podrán decidir si hacen público la creación de dichos NFT para ser minteados por los usuarios o si estarán ocultos y únicamente si ellos serán capaces de crearlos y otorgarlos. Estas entidades «emisoras» serán creadas con un usuario que no represente un individuo particular si no la entidad en si, estos serán capaces de abrir un *whitelist* sobre usuarios particulares para poder hacer uso de su *dashboard* internos.

Los *dashboards* internos de las entidades serán creados en paralelo con su respectiva entidad por medio de un mecanismo parecido al de «flavors» de Android. Esto se puede interpretar como una variante compilada para cada entidad personalizando así colores, imágenes, logos,etc.

El otro grupo que conformará nuestro modelo son los usuarios estos particulares estos serán los que se encargarán de interactuar con los *dashboards* generados para las entidades «emisoras» minteando así los NFT publicados.

Con esto mencionado se toma en cuenta que para la generación de estos usuarios/entidades y de los usuarios ordinarios se necesitaría programar la lógica de una billetera única para cada persona debido a que esto amerita un nuevo proyecto. Se utilizará Phantom como la billetera para demostrar las funcionalidades, utilizar el prototipo y realizar las pruebas. Esta decisión se basa en la popularidad y constante soporte que posee Phantom como una de las principales billeteras de solana. Para poder llevar a cabo dicho modelo se necesita definir un flujo de cómo será el proceso de verificación.

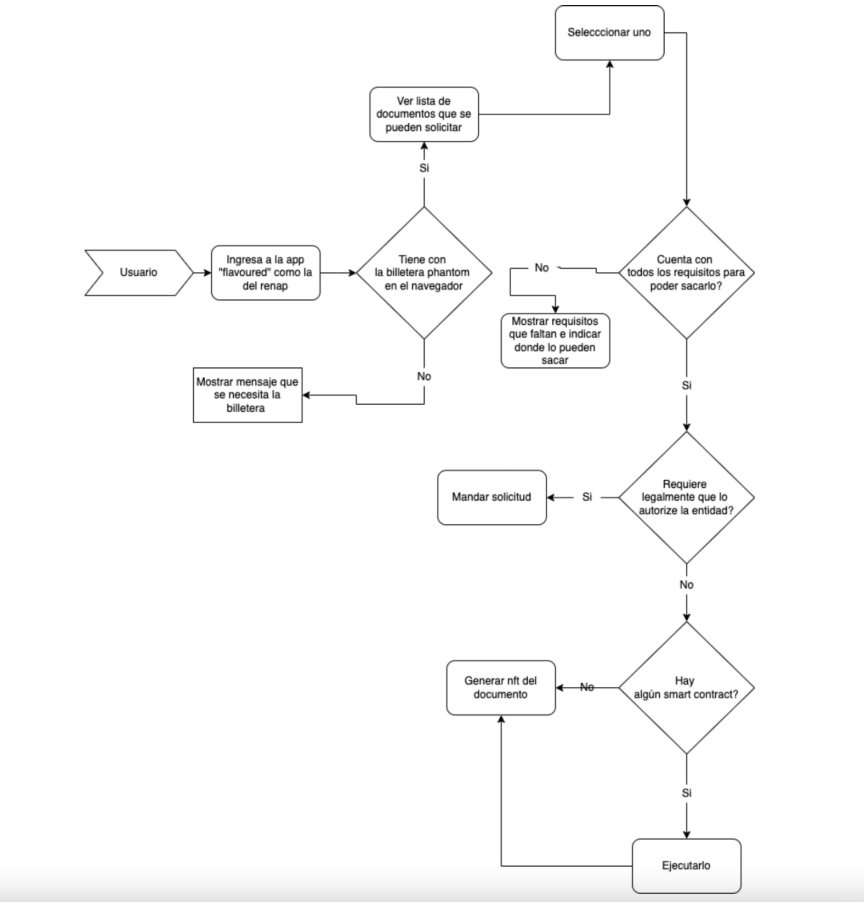


Figura 12: Proceso de usuario interactuando con un dashboard

Como se puede observar en la Figura 12 se espera que los usuarios ingresen a estos *dashboards* generados sea un proceso bastante simple con el que pueda generar un NFT dependiendo de si cumple con todos los requisitos para poder solicitarlo. Debido a que hay documentos que requieren una aprobación de una entidad dentro del proceso como requisitos legales se implementará un proceso donde las solicitudes sean representadas virtualmente con NFT para ser contempladas dentro de los requisitos. Dentro de la emisión de un documento se ejecutarán Smart Contracts si hay alguno.

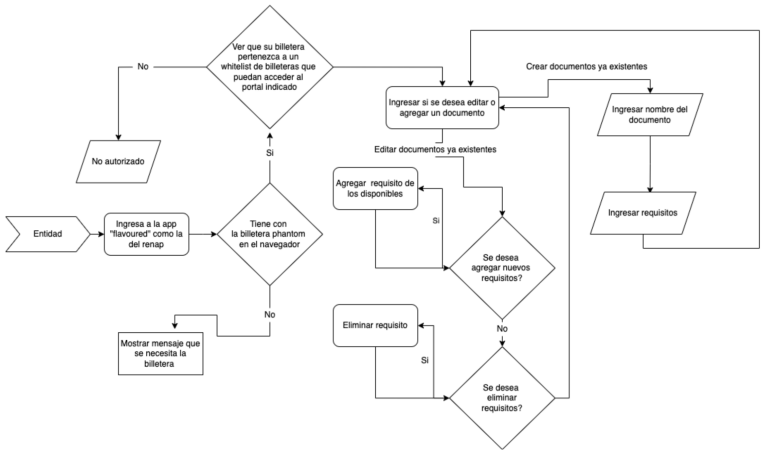


Figura 13: Ejemplo proceso usuario creando/editando un NFT

En el proceso para crear o editar un NFT o un documento se verificará primero si dicho usuario tiene permitido ingresar a el *dashboard* de la entidad y una vez adentro se permitirá agregar/eliminar requisitos o crear un nuevo documento con sus requisitos una vez finalizado este proceso se permitirá el desarrollo de diferentes acciones a ejecutar una vez se complete algún criterio para tenerlo como un *smart contract*.

Para este proyecto se tomarán diferentes fases de desarrollo: diseño, implementación, producción, documentación. Para la fase de diseño se realizará las normas de cómo deberá ser la estructura de los NFTs a producir para permitir la interacción entre ellos y la compatibilidad con la ejecución de *smart contracts* bajo ciertos criterios. El diseño será el que nos marcará la pauta de cómo crear estos NFT a través del *dashboard* para las entidades para que cualquier otra persona pueda implementar y hacer uso de los mismos.

En la fase de implementación se realizará el código para poder llevar a cabo el prototipo, se buscará demostrar la funcionalidades que se ofrecen por medio de un grupo pequeño de documentos que tengan funcionalidad en un ambiente real. Se decide únicamente simular esto debido a que la implementación de este programa requería tener digitalizado muchos datos para poder migrarlos a la plataforma lo cúal no se puede realizar debido a que ahorita se está pasando por una fase de digitalización a aplicaciones web en Guatemala y esperar una tercera migración sin aún completar este proceso no será algo realista. Para esta fase se utilizara de una combinación de VueJS + TailwindCSS para la creación de las aplicaciones web, durante el desarrollo de esta aplicación para poder interactuar con la red de solana se utilizará la librería oficial de solana de web3js que nos permite establecer la conexión entre el blockchain por medio de el protocolo de JSON-RPC 2.0 y HTTP requests (JSON RPC API, n.d.).

Para la creación de los NFT se utilizara una herramienta como Metaplex que es un protocolo desarrollado en Solana que nos permite crear/mintear NFT y visualizar los mismos a través de diferentes billeteras y aplicaciones se decidió utilizar este protocolo para ya que cuenta con una diversidad de herramientas para los desarrolladores en el manejo de los NFT permitiendo crear colecciones, juegos, tiendas de NFT y mucho más que cuenta con una gran comunidad detrás ayudando a mejorar la seguridad. Por último el desarrollo del código para los *smart contracts* y el resto de programas que correrán en el blockchain será en RUST debido a su opción de el *framework* anchor que nos brinda mayor facilidad para el desarrollo.

Durante esta fase de implementación se comenzará con el desarrollo de el UI que se utilizara para nuestro proyecto siendo en su mayor parte la creación de componentes para su reutilización una vez

completado esto se continuará con la interacción de la billetera de Phantom y el uso de la librería de web3js/solana. Una vez se logre una conexión exitosa se procederá al desarrollo de la creación y publicación de los NFTs de parte de las entidades con la ayuda de herramientas como Metaplex.

El siguiente paso durante esta fase es programar el *dashboard* en el cual los usuarios ingresarán para generar sus NFT y cumplirán con el proceso para poder generarlos y demostrar cómo este proceso de verificación puede cumplirse con los NFT para emitir más pero no se limita a eso. Durante el desarrollo de todos estos pasos se acompañarán de pruebas para asegurarnos su correcto uso y escalabilidad.

La fase de producción consistirá en montar este prototipo a una instancia de AWS para el uso del público en simulación de un ambiente real recopilando recomendaciones y cambios a realizar al modelo.

Por último estará la fase de documentación en la cuál se generará la documentación necesaria para que cualquier persona que desee implementar el prototipo pueda llevarlo a cabo además de un pequeño SDK facilitando la implementación del proceso de validación en otros casos de uso como lo podría ser un descuento en un gimnasio para todos los estudiantes de la UVG.

6.2. Desarrollo

A partir de esta sección nos referiremos a una "credencial modelo" como el NFT creado por una entidad que representa algún atributo de la identidad de una persona como el DPI y una "credencial de usuario" como el NFT generado a partir de la credencial modelo.

6.2.1. Desarrollo con el diseño inicial

Se inicio el desarrollo con el *dashboard* para la creación de credenciales modelo para las entidades. El *framework* de CSS que se decidió utilizar fue el mismo que el que se propuso en el diseño TailwindCSS permitiéndonos configurar nuestra paleta de colores dentro de la app, para la navegación se optó por implementar Vue Router y Vuex para el manejo de nuestro estado. Por default al crear una nueva aplicación con el CLI de VueJs nuestro module *bundler* es Webpack 5. Una vez finalizado nuestro *setup* se procedió a desarrollar el flujo de ingreso a la aplicación por medio de chequear si la *public key* de la billetara esta autorizada o no en un *whitelist* que la aplicación posee. El flujo de ingreso se comporta de igual manera al mencionado en la Figura 13. Durante esta fase se completo la simulación de todo el proceso con datos falsos para asegurarnos que el UI de la aplicación estuviera completo y únicamente faltará su integración con el *smart contract* para generar las credenciales modelo.

Durante este proceso se inicio utilizando Anchor como *framework* de Rust para el desarrollo de *smart contracts* y los *crates* de Rust que provee Metaplex para la implementación de su programa *Token Metadata* durante este período se encontró inconvenientes al momento de estar implementándolo con la aplicación desarrollada en VueJs debido a la falta de documentación en como utilizar sus métodos y falta de descripción en los errores al momento de ejecutar la transacción lo cual atrasaba su desarrollo debido a que para cada cambio era necesario hacerle *deploy* a nuestro programa en la *devnet* de Solana.

6.2.2. Cambio a SDK en JS de Metaplex

Debido a los inconvenientes de la integración de nuestro *smart contract* utulizando Anchor y los *crates* de Metaplex se optó por utilizar el SDK en JS que Metaplex nos provee. Siendo este nuestro

primer cambio al diseño original.

Permitiéndonos una mejor experiencia y facilidad de desarrollo para la creación de las credenciales modelo y credenciales de usuario. Durante la implementación del SDK se topó con dos inconvenientes que comprometían el desarrollo e integridad del proyecto:

- Nuestro *Webpack 5* no lograba resolver las clases de Metaplex restringiéndonos de su uso.
- Nuestra llave privada con la que firmábamos las transacciones de parte de la entidad quedaba comprometida al ser expuesta en el código del *front-end*. Es decir si una persona inspeccionaba nuestro código en Chrome u otro navegador podría encontrar con facilidad la llave con la cual firmamos las transacciones. Con esto permitiéndole crear tanto credenciales modelo y credenciales de usuario a nombre de la entidad y habilitando la modificación de su metadata.

Solución a Webpack 5

Como solución al error que Webpack nos estaba generando en conjunto con el SDK de Metaplex se movió el proyecto a ViteJS. ViteJS es una herramienta que facilita el desarrollo en múltiples *frameworks* de front end en JavaScript enfocado en mayor facilidad de configuración para el proyecto, *bundless development* y velocidad.

Se decidió al igual que en *Webpack 5* poner la configuración recomendada por Metaplex para el proyecto en ViteJs.

6.2.3. Desarrollo utilizando el SDK en JS de Metaplex

En esta fase se terminó de implementar el modulo de las credenciales modelo para las entidades permitiendo lo siguiente:

- Creación de las credenciales modelo asignándoles una imagen, un identificador y su nombre.
- Habilitación de requisitos para poder generar una credencial de usuario.
- Expiración en credenciales modelo. Es decir si se indica que luego de 1 año expira cuando el usuario intente utilizar dicha credencial de usuario emitida a partir del modelo para verificarse dentro de un proceso saldrá rechazada.
- Edición de una credencial modelo permitiéndole cambiar sus requisitos.

El estándar de los NFT que representaran las credenciales modelo son los siguientes:

- El modelo debe de poseer un puntero URI ubicando a un JSON *off-chain* forzosamente.
- El JSON poseera un campo model con el valor de "*credential*" que nos indica que es una credencial válida para nuestro sistema.
- Campo *collection* vacío esto nos indica que es una credencial modelo

Las credenciales de usuario tendrán el mismo estándar a diferencia que el campo *collection* tendrá como valor la llave pública de la credencial modelo a la que pertenece.

Durante el desarrollo se plantearon nuevas incógnitas como el de como se irá a resolver el tema de la llave privada para asegurarla ahora que dependemos de este SDK en JS; si el imprimir nuevas ediciones de una credencial modelo e intentar crear un NFT que sirva como colección era la mejor opción para emitirlos como credenciales a los usuarios.

Cambio a *Candy Machines*

Se decidió utilizar el módulo de *Candy Machines* en Metaplex para que maneje la logística de publicación, control de precio y asignación a una colección al generar un NFT esto debido a que este programa que Metaplex nos ofrece cuenta ya con una gran cantidad de opciones para personalizar y configurar la manera en que una determinada cantidad de NFT'S se facilita. Nos podemos imaginar estos como dispensadores en los cuales se decide el momento en el cual serán visibles al usuario, parte de que colección perteneces, regalías por la venta del NFT, sus creadores, etc. Esto nos beneficia en que nos reduce la cantidad de validaciones que hay que realizar sobre el *dashboard* que emitirá documentos a los usuarios además de reducirnos la cantidad de errores que se puedan generar al crear nuevas ediciones de un NFT que fue la primera alternativa y una mayor escalabilidad en conjunto con el proyecto de Metaplex en sus actualizaciones.

Se habilitó un nuevo módulo dentro de nuestra aplicación para almacenar las *Candy Machines* creadas como dispensadores. En estas se decidió permitir al usuario decidir que precio tendría cada NFT dentro del dispensador y a que colección pertenecía es decir elegir una de las credenciales modelo como indicador de colección.

Convertir la aplicación *permissionless*

Durante las pruebas que se realizaron con los diversos usuarios nos indicaron que si había un proceso de solicitud no notaban una diferencia en muchos procesos de hoy en día en donde la entidad tiene el control absoluto sobre la decisión en un proceso perdiendo transparencia en el proceso. Debido a esto se decidió optar por un sistema completamente *permissionless* eliminando la opción de crear solicitudes para emitir un documento por parte de un usuario y únicamente necesitando cumplir con los requisitos de dicha credencial.

6.2.4. Desarrollo de *dashboard* para los usuarios

El desarrollo de la aplicación visible a los usuarios siguió el flujo definido en la Figura 12. Pero con el mismo inconveniente de que la llave privada quedaba expuesta con el fin de poder firmar las transacciones. En esta fase de desarrollo se implemento la lógica que conllevaría el validar los requisitos de una credencial antes de emitirlos y si una credencial se encuentra expirada o no que. Dicho proceso presento nuevos desafíos:

- Si la validación de requisitos y fecha de expiración se realiza *off-cahin* un usuario con acceso a la llave privada la cual es fácil de acceder por ahora podría burlar esta validaciones.
- La validación de la caducidad de una credencial de usuario tendrá que lograrse únicamente con la metadata de la credencial modelo. Es decir el intentar agregar un *timestamp* luego de emitir la credencial de usuario en su metadata abre la posibilidad que ocurra un error en medio del proceso obviando así su fecha de expiración y permitiendo que quede incompleto.
- Debido a que el tiempo que tarda en cargar todos los NFT junto a su metadata y transacciones puede ser relativamente tardado es necesario cachear los resultados, el desarrollo de web nos permite utilizar el *local storage* lo cual nos cumple con el requisito de cachear y reducir tiempos pero este es muy inseguro ya que un usuario puede acceder a el desde consola y modificar sus credenciales serializadas logrando saltarse validaciones.

Solución a la validación de expiración

Ya que toda interacción dentro del blockchain de solana necesita transacciones podemos obtener la fecha de creación de un NFT sin necesidad de tener que agregarlo a su metadata.

Con esto en mente se decidió que las credenciales de usuario se estarán importando con sus transacciones tomando la primera como su creación y comparándola con el tiempo de expiración que se le configuro a la credencial modelo.

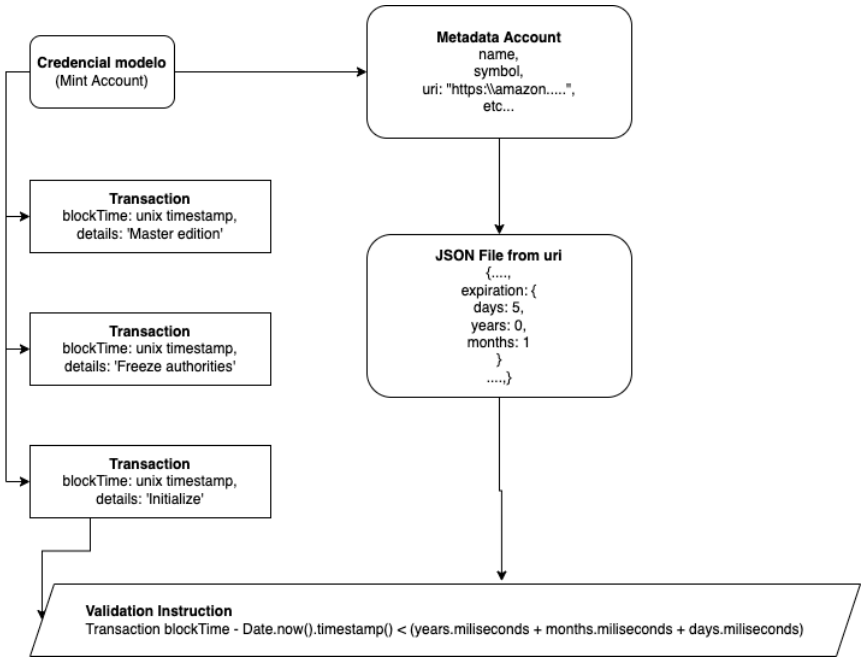


Figura 14: Flujo para validación de fecha de expiración

En la Figura 14 observamos que con el JSON que obtenemos a través de nuestro *metadata account* y la primer transacciones podemos verificar si la credencial se encuentra expirada o no procesando todo en milisegundos.

Implementación de ExpressJS

Observando diversos *market places* de NFT como Solanart u OpenSea observamos que se apoya en en un api propio para obtener la información de los NFT y las acciones como crear una subasta sobre un NFT o incluso comprarlo.

Inspirándonos en estos dos *market places* que se han establecido entre los más populares dentro de la comunidad de Solana e incluso Ethereum en el caso de OpenSea se decidió mover todo el código que involucrara interactuar con la red de Solana a un api utilizando Node16 y ExpressJs.

Una vez finalizada la implementación en ExpressJS se utilizó el paquete de memory-cache en npm para cachear las credenciales de usuario de determinado *Account* y las credenciales modelo obteniendo una mejor experiencia de usuario como resultado y mayor escalabilidad en cuando a su uso. Ya que los NFT creados por una *Candy Machine* son inmutables de Metaplex no necesitamos estar monitoreando que las credenciales de usuario puedan verse alteradas únicamente que se se agreguen nuevas credenciales a la lista.

Debido a que toda interacción del blockchain que requiera una firma se realiza por medio del api podemos remover la llave privada ambos *dashboards* y tenerlo en una ubicación segura dentro del servidor que sirva de *host* para el api resolviendo el problema originado en el cambio al SDK de JavaScript.

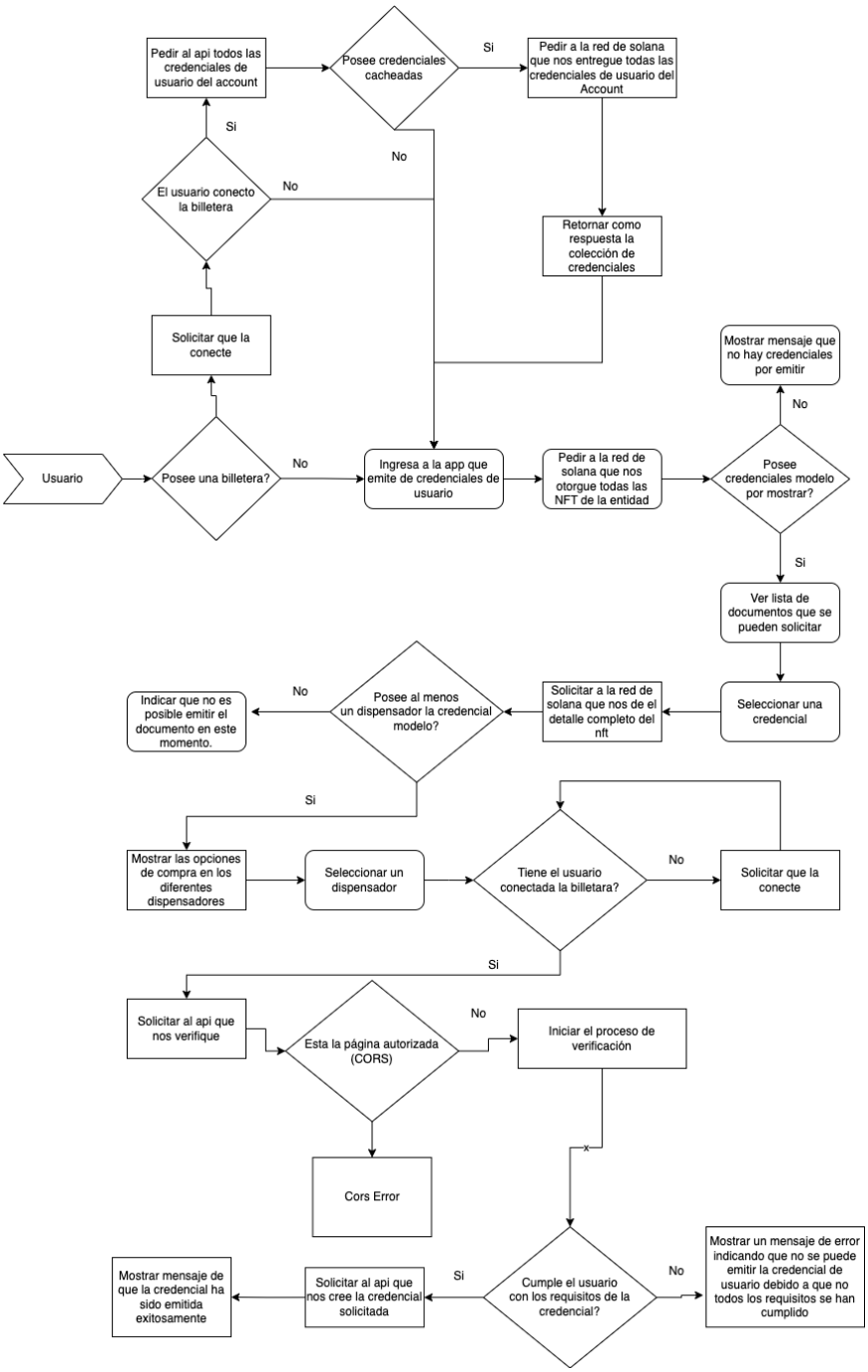


Figura 15: Proceso final de usuario interactuando solicitando una credencial de usuario

6.3. Implementación

Esta fase se enfoca en cómo logramos llevar nuestro proyecto de un ambiente local a un ambiente de producción. Tanto como las aplicaciones web como para nuestro api.

6.3.1. Aplicaciones web

Se decidió utilizar Github Pages como *host* de nuestra página debido a costos del mismo y su facilidad para poderlo llevar a cabo. En el proceso se complicó la compilación con Vite JS. Esto debido a que en su fase de optimización no estaba compilando de manera correcta la herencia de una clase en la librería de *@solana/web3.js*. Se decidió subir un *issue* en Github en el cual mencionaron que en la nueva versión de *Metaplex* el error con Webpack 5 se había solucionado.

Debido a esto se regresó a utilizar Webpack 5 para compilar nuestro proyecto y subirlo a producción.

6.3.2. Api

Para la implementación del api se utilizó una instancia de AWS con Ubuntu 20.4. Dentro de la misma se configuró Ngnix como *proxy* para poder transmitir nuestros HTTP (80) y HTTPS (443) *requests* a el puerto 3000 en el que se encuentra nuestra aplicación escuchando a través de PM2; se tomó la decisión de implementarlo para poder tener una mayor escalabilidad en el api.

CAPÍTULO 7

Resultados

7.1. Costos

Todos los costos son calculados con base en poder cumplir con las necesidades de una entidad como el RENAP.

7.1.1. Costos arquitectura prototipo

Nombre	**Descripción**	**Precio mensual (Q)**
t3.small	Instancia de AWS, 2GB de memoria, 2vCPUs	108.00
EBS	40GB General Purpose SSD (gp2)	40.00
Snapshots	2 snapshots diarios	65.50
S3	Servicio de Amazon para almacenar la metadata de los NFT *.	80.00
Total		Q 327.00

Cuadro 1: Desglose backend
* Costo calculado en 20GB al mes de almacenamiento mensual, 50,000 requests mensuales solicitando información a amazon S3, 50,0000 requests almacenando información a S3, 100 GB de información transferida fuera de la región de US East (Ohio).

Nombre	Descripción	Precio mensual (Q)
t3.small	Instancia de AWS, 2GB de memoria, 2vCPUs	142.80
EBS	25GB General Purpose SSD (gp2)	24.00
Snapshots	1 snapshots diarios	36.50
Total		Q 203.30

Cuadro 2: Desglose costos dashboard interno

Nombre	Descripción	Precio mensual (Q)
t3.small	Instancia de AWS, 2GB de memoria, 2vCPUs	142.80
EBS	25GB General Purpose SSD (gp2)	24.00
Snapshots	1 snapshot diario	36.00
Total		Q 203.30

Cuadro 3: Desglose costos dashboard para usuarios

Nombre	Descripción	Cantidad	Precio unitario mensual (Q)	Total mensual (Q)
Servidor Backend	Servidor para el api que servira como backend de ambas plataformas	1	327.00	327.00
Dashboard interno	Servidor para la aplicación del dashboard interno.	1	203.30	203.30
Dashboard usuarios	Servidor para la aplicación del dashboard para que los usuarios puedan emitir sus documentos.	1	203.30	203.30
Creación de credenciales	Costo aproximado de creación de diferentes credenciales. Esto es únicamente su formato.	100	0.12	12.00
Creación de dispensadores	Costo aproximado de creación de diferentes dispensadores con 15000 credenciales cada uno	6	5,778.80	34,672.00
Desarrolladores Senior	Desarrolladores que se encargarán de liderar el desarrollo y asegurar la estabilidad de las aplicaciones.	4	20,000.00	80,000.00
Desarrolladores junior	Desarrolladores encargarán de bugfixing y desarrollo de pequeñas features.	4	8,000.00	32,000.00
Total				Q 147,417.30

Cuadro 4: Costo arquitectura prototipo

7.1.2. Costos arquitectura actual

Nombre	Descripción	Cantidad	Precio unitario mensual (Q)	Total mensual (Q)
Operador Registral I	Segundo nivel de operadores registrales primer nivel.	1,096	3,850.00	4,219,600.00
Operador Registral II	Segundo nivel de operadores registrales segundo nivel.	36	4,050.00	Q145,800
Operador Registral III	Segundo nivel de operadores registrales tercer nivel	15	4,250.00	63,750.00
Operador Registral IV	Segundo nivel de operadores registrales cuarto nivel	14	5,250.00	73,500.00
Operador Registral V	Segundo nivel de operadores registrales cuarto nivel	3	6,750.00	20,250.00
Total				Q4,522,900.00

Cuadro 5: Costo operadores registrales
* Toda la información recopilada de la nómina del RENAP se obtuvo de: Nómina RENAP

Nombre	Descripción	Cantidad	Precio unitario mensual (Q)	Total mensual (Q)
Guardián	Guardián oficinas registrales y auxiliares.	202	3,435.00	693,870.00
Auxiliar de limpieza I	Auxiliar de limpieza primer nivel.	54	3,435.00	185,490
Auxiliar de limpieza II	Auxiliar de limpieza segundo nivel	4	3,550.00	14,200.00
Auxiliar de almacén I	Auxiliar de almacén primer nivel	8	3,750.00	30,000.00
Auxiliar de almacén II	Auxiliar de almacén segundo nivel	2	4,250.00	9,500.00
Auxiliar de almacén III	Auxiliar de almacén tercer nivel	1	8,250.00	8,250.00
Total				Q774,320.00

Cuadro 6: Mantenimiento oficinas registrales
* Toda la información recopilada de la nómina del RENAP se obtuvo de: Nómina RENAP

7.2. Velocidad

7.2.1. Velocidad de prototipo

Nombre de proceso	Tiempo promedio (s)
Creación de credencial	15.50s
Creación de dispensador	17.27s
Emisión de credencial	13s
Verificación de credencial	2.2s

Cuadro 7: Tiempos actividades en prototipo

7.2.2. Velocidad de emisión credenciales arquitectura actual

Nombre de proceso	Tiempo promedio
Entrega de DPI	30 días hábiles
Entrega certificado de nacimiento (digital)	15s
Constancia de matrimonio (tramite inicial)	30 minutos

Cuadro 8: Tiempos actividades en RENAP

7.2.3. Velocidad de validación de credenciales

Credencial	Descripción	Tiempo promedio
DPI	Verificación dactilar o por medio de verificación facial realizado contra pago.	N/A
Certificado de nacimiento	Verificación por medio de Código QR	15s
Constancia de matrimonio	Verificación por medio de Código QR	15s

Cuadro 9: Validaciones otorgadas por el RENAP
*Estas validaciones no permiten la interoperabilidad entre aplicaciones.

7.3. Escalabilidad

7.3.1. Pruebas de estrés

Nombre de proceso	Tiempo promedio (s)
Creación de credencial	16.50s
Creación de dispensador	17.7s
Emisión de credencial	13.2s
Verificación de credencial	2.1s

Cuadro 10: Tiempos con 10 solicitudes simultaneas

Nombre de proceso	Tiempo promedio (s)
Creación de credencial	22.50s
Creación de dispensador	21.7s
Emisión de credencial	16.2s
Verificación de credencial	2.1s

Cuadro 11: Tiempos con 100 solicitudes simultaneas

7.4. Encuestas

7.4.1. Utilidad del SDK

Las encuestas se realizaron en un total de 23 programadores de los cuales 15 de ellos no poseen ni un tipo de experiencia utilizando blockchain y 8 si han desarrollado en blockchain.

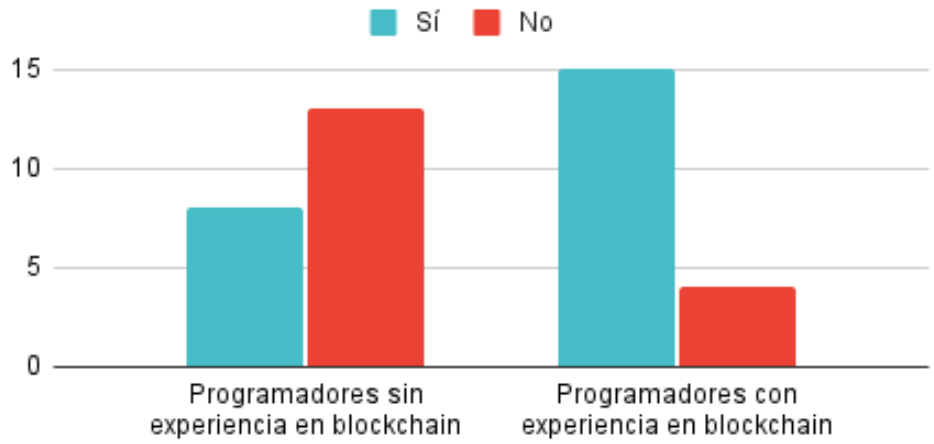


Figura 16: ¿Ha escuchado de solana?

Figura 17: ¿Consideraría útil un SDK para el desarrollo?

Figura 18: ¿Desarrolla en Javascript?

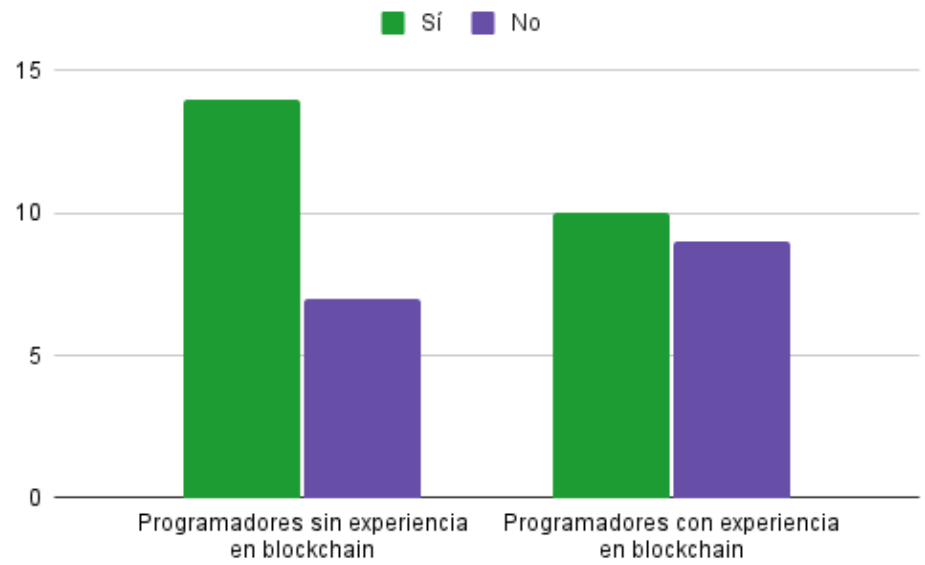


Figura 19: ¿Considera fácil de utilizar el SDK “i-am-verifiable-button”?

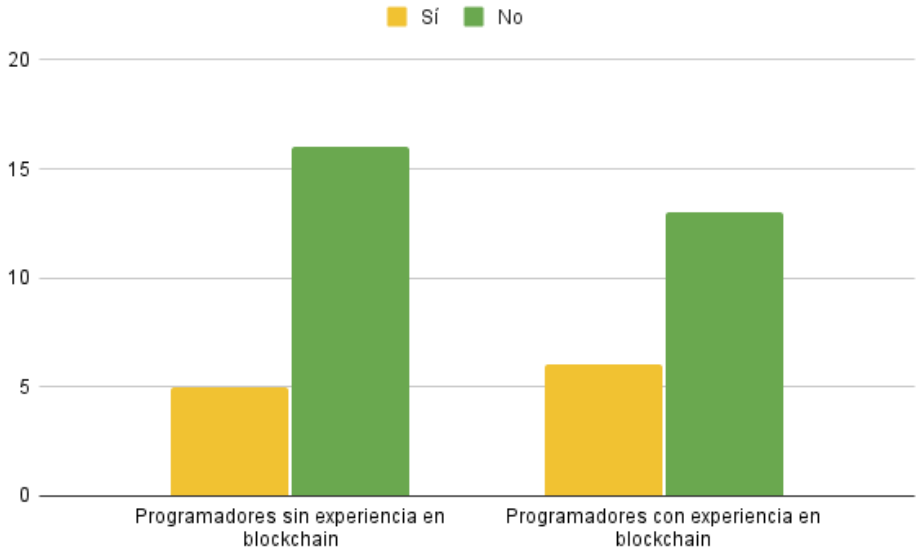


Figura 20: ¿Considera que para utilizar el SDK“i-am-verifiable-button” es necesario tener expeciencia en blockchain?

7.4.2. Encuestas a trabajadores del gobierno

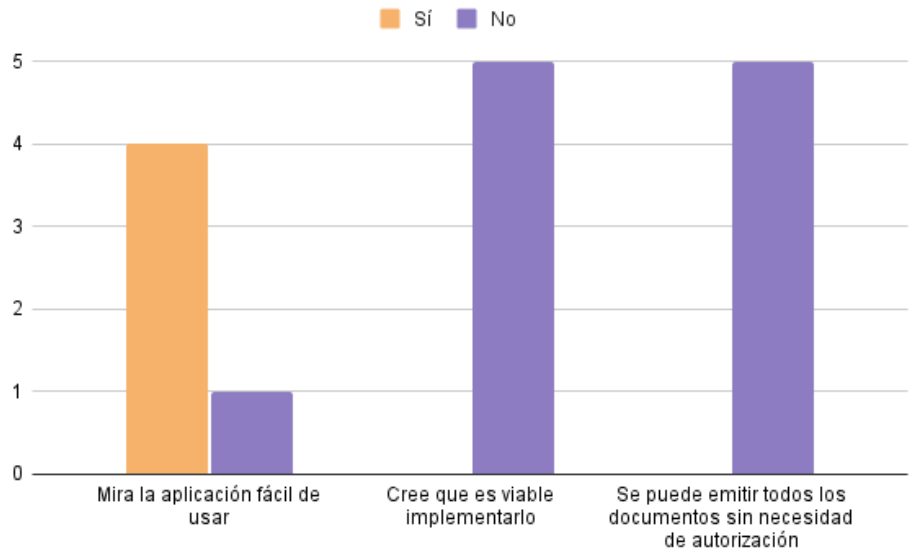


Figura 21: Resultados de encuestas a trabajadores

Comentarios

Dentro de las encuestas se pidió favor de comentar si la respuesta de las preguntas "Cree que es viable implementarlo" y "Se pueden emitir todos los documentos sin necesidad de autorización". Las respuestas comunes fueron las siguientes:

- El proceso de digitalización de los datos llevará años y actualmente se esta realizando pero para integrarse con la tecnología web 2.0.
- Por cuestiones legales es necesario la firma y autorización de ciertas entidades al emitir documentos. No se puede esperar a que la ley cambie a favor de un programa.

7.5. Prototipos

Dashboard de emisión de documentos
Dashboard que se utiliza para que los usuarios puedan emitir sus credenciales.

Dashboard interno
Dashboard que se utiliza para crear las credenciales modelo.

Landing Page
Landing page que permite utiliza el SDK generado con mayor facilidad.

SDK
SDK para verificar credenciales por medio de un botón.

CAPÍTULO 8

Conclusiones

1. El tiempo promedio de verificación una credencial basada en NFT es más rápido que su competencia
2. El tiempo promedio de creación una credencial basada en NFT es más rápido con la arquitectura propuesta vs la actual.
3. La validación de múltiples credenciales para un solo proceso no influye en el tiempo de validación.
4. La validación de credenciales con fechas de expiración si afectan el tiempo del mismo.
5. La arquitectura propuesta ayudaría a reducir costos para la emisión de documentos aunque aún se debería de mantener personal para la gente sin acceso a internet.
6. El desarrollo del SDK para poder verificar credenciales facilita y motiva a los desarrolladores a utilizar el modelo basado en NFT.
7. La idea de utilizar una arquitectura basada en NFT no es viable, el proceso de digitalización de todos los documentos aun no se encuentra completo.

CAPÍTULO 9

Recomendaciones

1. Utilizar uno de los ejemplos dados por parte del equipo de Metaplex como punto de inicio al proyecto para evitar inconvenientes en su desarrollo.
2. Apoyar el desarrollo del *dashboard* con un api que se encargue de recibir los nfts, mantener un control, recopilar más información, etc.
3. Desarrollar una billetera específica para los usuarios que sirva como identificador inicial.
4. Habilitar un modulo para permitir el *mint* de credenciales y ocultarlas del usuario habilitando así únicamente que la entidad pueda decidir a quien entregarlo.
5. Iniciar el proyecto leyendo acerca de Solana Token Program para comprender las bases de como funciona sus tokens.
6. Implementar un sistema de autenticación extra en el módulo de generación de documentos.
7. Robustecer el api con una base de datos en el que podamos mantener mayor control, registrar más información, dar mas información acerca de los patrones de consumo de los nfts, validaciones exitosas, validaciones fraudulentas, etc.
8. Utilizar un proveedor privado para el programa para mejorar tiempos de interacción con la red de Solana.
9. Buscar que documentos se pueden emitir por medio de la plataforma que no requieran autorización de una entidad por parte de la ley.

CAPÍTULO 10

Bibliografía

101 Blockchains. (2021). *What Is DLT (Distributed Ledger Technology) ?* Consultado el 17 de octubre de 2022, desde https://101blockchains.com/what-is-dlt

Ashforth, B. E., & Mael, F. (1989). Social Identity Theory and the Organization. *14*. https://doi.org/10.2307/258189

Binance. (2022). *Blockchain*. Consultado el 11 de octubre de 2022, desde https://academy.binance.com/en/glossary/blockchain

C. Marlene Fiol, E. J. O. (2005). Identification in Face-to-Face, Hybrid, and Pure Virtual Teams: Untangling the Contradictions. Organization Science. *16*. https://doi.org/10.1287/orsc.1040.0101

Grolleau, G., Lakhal & Mzoughi, N. (2008). An Introduction to the Economics of Fake Degrees. *Journal of Economic Issues*, *42*, 673-693. https://doi.org/10.1080/00213624.2008.11507173

IBM. (2022). *Smart contracts defined*. Consultado el 18 de octubre de 2022, desde https://www.ibm.com/topics/smart-contracts

Metaplex. (2022). *Token Metadata Overview*. Consultado el 12 de octubre de 2022, desde https://docs.metaplex.com/programs/token-metadata/overview

Milanov, E. (2009). The RSA Algorithm [(Visitado el 22/10/2022)].

San Jose State University. (2022). The Mathematics behind RSA [(Visitado el 22/10/2022)].

Shah, M., Aditya, A., Bindra, D., Omkar, V. S., & Seervi, A. (2020). Providing a Secure, Reliable and Decentralized Document Management Solution Using Blockchain by a Virtual Identity Card. *14*. https://doi.org/10.5281/zenodo.3669214

Slomovic, A. (2014). Privacy Issues in Identity Verification. *12*. https://doi.org/10.1109/MSP.2014.52

Solana. (2022a). *Accounts*. Consultado el 12 de octubre de 2022, desde https://docs.solana.com/developing/programming-model/accounts

Solana. (2022b). *SOLANA Explorer*. Consultado el 18 de octubre de 2022, desde https://explorer.solana.com

Solana. (2022c). *Solana Rpc*. Consultado el 18 de octubre de 2022, desde https://solana.com/rpc

Solana Cookbook. (2022). Program Derived Addresses (PDAs). Consultado el 18 de octubre de 2022, desde https://solanacookbook.com/core-concepts/pdas.html#generating-pdas

Summers, A. (2022). *Understandind blockchains and criptocurrencies*. CRC Press.

Yakovenko, A. (2022). *Solana: A new architecture for a high performance blockchain v0.8.13*. Consultado el 18 de octubre de 2022, desde https://solana.com/solana-whitepaper.pdf

CAPÍTULO 11

Glosario

epoch: El tiempo, que es el número de slots por el cual el horario de validadores es válido en la red. 15

lamports: Token nativo en solana que representa 0.000000001 sol. 15

metadata: Conjunto de datos que da más información acerca de otro dato. 15

mintear: Se denomina al proceso de publicar de manera única un token dentro de una red de blockchain para permitir que este sea comprable (Martin, n.d.).. 19

no fungible: Representa que no puede ser sustituido por otros de la misma especie, validad y cantidad. 16

PDA: Program Derived Account. 16, 17, 19

RPC: Los *Remote procedure calls* son una técnica para crear sistemas distribuidos. Permite a un programa en una maquina llamar a una subrutina en otra maquina sin conocer que es de manera remota. Más que un protocolo de transporte es un método de utilizar sistemas de técnicas de comunicación existentes en una manera transparente. 15

RSA: Rivest–Shamir–Adleman. 10, 11

Smart Contracts: Según IBM un smart contract se le denomina a un programa que esté guardado en el blockchain que corre cuando ciertas condiciones se cumplan. Usualmente se utilizan para automatizar la ejecución de un acuerdo en el que todos los participantes puedan estar seguros del resultado sin ni un intermediario.. 21

token: Divisa o activo en el mundo de la criptografía. 16

TPS: Transacciones por segundo. 1