# Solana: Un Análisis de su Arquitectura y Escalabilidad en la Web3

*Jaime Muñoz*

*Universidad de Antioquia*

*Jaime.munozq@udea.edu.co*

## Introducción

En la mayoría de las blockchains públicas, cada nodo utiliza su propio reloj local sin tener conocimiento de los relojes de otros participantes, lo que genera incertidumbre al aceptar o rechazar mensajes con base en sus marcas de tiempo. Solana propone una solución llamada ***Proof of History (PoH),*** que crea un registro de eventos con una verificación confiable del paso del tiempo. Esto permite a los nodos confiar en la secuencia temporal registrada en el ledger, lo que mejora el ordenamiento de mensajes y la sincronización en la red sin necesidad de confianza externa [1].

El ***Proof of History (PoH)*** es un mecanismo criptográfico utilizado en Solana para verificar el paso del tiempo entre dos eventos, sin depender de relojes externos. Funciona ejecutando una función criptográficamente segura, donde el resultado no puede predecirse a partir de la entrada, y debe ser completamente calculado para generar una salida válida.

Este proceso sigue los siguientes pasos:

- Ejecución secuencial: La función se ejecuta en secuencia en un solo núcleo de CPU, donde cada salida generada sirve como entrada para la siguiente operación. Esta cadena continua de cálculos crea una secuencia verificable.

- Registro periódico: Durante la ejecución, la función registra periódicamente su salida y el número de veces que ha sido llamada. Estos registros permiten revalidar la secuencia.

- Verificación paralela: Aunque la función se ejecuta en un solo núcleo, los resultados pueden verificarse en paralelo por otros nodos de la red. Cada segmento de la secuencia puede ser revisado de manera independiente por múltiples núcleos, lo que facilita la validación rápida.

- Marcas de tiempo: Los datos pueden añadirse a esta secuencia de tiempo mediante la inclusión de los datos o un hash dentro del estado de la función. Esto genera una marca de tiempo que garantiza que los datos fueron creados antes de que se generara el siguiente hash en la secuencia.

- Escalabilidad horizontal: El diseño de ***PoH*** soporta la escalabilidad horizontal. Múltiples generadores pueden sincronizarse entre sí mezclando sus estados dentro de las secuencias de los otros, permitiendo que diferentes nodos trabajen en paralelo.

Solana se ha diseñado para el uso generalizado y masivo al ser eficiente en términos de energía, extremadamente rápida y muy económica. Los creadores de Solana, como ***Anatoly Yakavenko***, provienen del ámbito de redes de telefonía móvil, lo que les ha permitido enfocarse en la escalabilidad y la eficiencia. Estas características son fundamentales para que el público adopte y utilice la tecnología blockchain de manera más accesible [2].

A diferencia de otras blockchains que dependen de procesos de minería intensivos en energía y tiempo, Solana utiliza un mecanismo de validación llamado ***proof of stake,*** eliminando la necesidad de minería. Además, introduce la innovación del ***proof of history,*** que le permite validar transacciones de forma aún más rápida. Esto no solo reduce drásticamente el consumo energético, haciéndolo comparable al de unas pocas búsquedas en Google, sino que también reduce las tarifas de transacción a fracciones de centavo, en contraste con otras redes blockchain donde los costos pueden alcanzar cientos de dólares por transacción [8].

Así pues, Solana no solo ofrece una infraestructura altamente escalable y eficiente, sino que también redefine los límites de la tecnología blockchain al proponer soluciones innovadoras como Proof of History. Esta combinación de velocidad, bajo costo y sostenibilidad energética hace que Solana sea una de las blockchains más prometedoras para aplicaciones descentralizadas (dApps), finanzas descentralizadas (DeFi), y otros casos de uso que requieren un alto rendimiento. Su capacidad para soportar miles de transacciones por segundo (TPS) posiciona a Solana como un actor clave en la adopción masiva de blockchain, acercándonos a un futuro donde la tecnología descentralizada sea parte integral de la vida cotidiana.

## ¿Por qué solana?

Solana busca una solución distinta al "trilema de la blockchain" un concepto propuesto por Vitalik Buterin, el creador de Ethereum, que sugiere que es extremadamente difícil para una blockchain optimizar simultáneamente tres propiedades clave: **descentralización**, **seguridad** y **escalabilidad**. Esto lo logra a

través de su enfoque de consenso híbrido entre Proof of Stake (***PoS***) y Proof of History (***PoH***) [3].

***PoH*** es clave en este enfoque porque elimina la necesidad de que los nodos de la red se comuniquen constantemente para sincronizarse y acordar el paso del tiempo. Esto no solo reduce drásticamente la latencia, sino que permite a Solana escalar sin los cuellos de botella típicos de otras blockchains. Mientras que otras redes requieren confirmaciones de múltiples nodos antes de validar un bloque, Solana puede mantener una red distribuida que alcanza velocidades superiores a 50,000 TPS gracias a ***PoH***, al mismo tiempo que garantiza la integridad de la secuencia de eventos.

Este mecanismo también habilita a Solana para alcanzar tiempos de bloque significativamente más cortos (400 ms) en comparación con otros sistemas. De esta forma, Solana mantiene la seguridad y descentralización al tiempo que iguala la velocidad y capacidad de sistemas centralizados como los utilizados en transacciones bancarias o de tarjetas de crédito [4].

Por otro lado, en Solana, el líder que secuencia las transacciones se elige mediante ***PoS***, un mecanismo que selecciona nodos validadores con base en la cantidad de tokens que poseen y están dispuestos a "apostar" en la red. Este nodo líder organiza las transacciones y reduce la carga de comunicación entre los nodos. Esto mejora la escalabilidad, ya que coordinar menos nodos directamente permite que la red maneje más transacciones por segundo.

De esta manera, debido a que ***PoS*** elige un líder para secuenciar las transacciones y ***PoH*** permite que la red mantenga el orden de las transacciones de manera independiente, Solana puede manejar una cantidad significativamente mayor de TPS que muchas otras blockchains, incluso sin depender de una fuente centralizada de tiempo.

## Estructura de la red.

### *Cuentas.*

En Solana, todos los datos se almacenan en lo que se denomina "cuentas". La forma en que se organizan los datos en Solana se asemeja a un almacén de clave-valor, donde cada entrada en la base de datos se denomina "cuenta" [6].


**Ilustración 1. Estructura de las cuentas en solana**

Cada cuenta se puede identificar por su dirección única, representada como 32 bytes en el formato ***Ed25519.***

***Ed25519*** utiliza la matemática de curvas elípticas para generar pares de claves (pública y privada). La clave privada se utiliza para firmar un mensaje, produciendo una firma digital que puede ser verificada por cualquier persona con la clave pública correspondiente. La seguridad del sistema se basa en la dificultad de resolver el problema del logaritmo discreto en el contexto de las curvas elípticas [14].

La dirección pública es el identificador único de la cuenta.

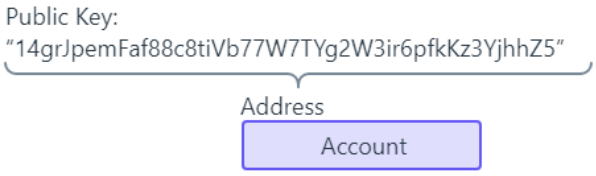


**Ilustración 2. Formato de la dirección pública de la cuenta**

Además, las cuentas tienen un tamaño máximo de **10 MB (**10 megabytes) y los datos almacenados en cada cuenta en Solana tienen la siguiente estructura conocida como ***AccountInfo [18].***

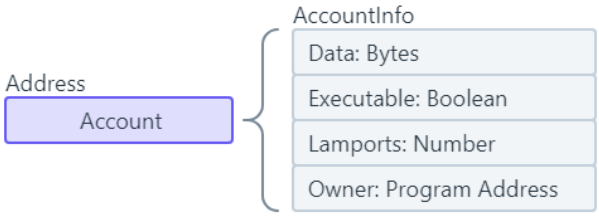


**Ilustración 3. Información de la cuenta**

Los campos que incluye son los siguientes:

- data: Una matriz de bytes que almacena el estado de una cuenta. Si la cuenta es un programa (contrato inteligente), almacena el código del programa ejecutable.
- executable: Un indicador booleano que indica si la cuenta es un programa.
- lamports: Una representación numérica del saldo de la cuenta en lamports , la unidad más pequeña de SOL (1 SOL = 1 mil millones de lamports).
- owner: Especifica la clave pública (ID del programa) del programa que posee la cuenta. Solo el programa dueño de la cuenta puede modificar sus datos o deducir su balance de lamports. Sin embargo, cualquiera puede aumentar el balance.

Solana contiene un pequeño puñado de programas nativos que son parte de la implementación de los validadores y proporcionan varias funcionalidades principales para la red. Por defecto, todas las cuentas nuevas pertenecen al programa del sistema. El Programa del Sistema realiza varias tareas clave como:

- Creación de cuentas nuevas: El programa del sistema es el único que puede crear cuentas nuevas.

- Asignación de espacio: Establece la capacidad de bytes para el campo de data de cada cuenta.
- Asignar propietario a una cuenta: Una vez que el programa del sistema crea una cuenta, puede reasignar al propietario designado a una cuenta diferente. Así es como los programas personalizados asumen la propiedad de cuentas creadas por el programa del sistema.

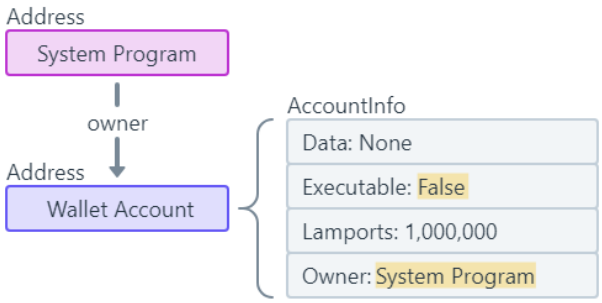


**Ilustración 4. Ejemplo de una cuenta en solana que no es un programa**

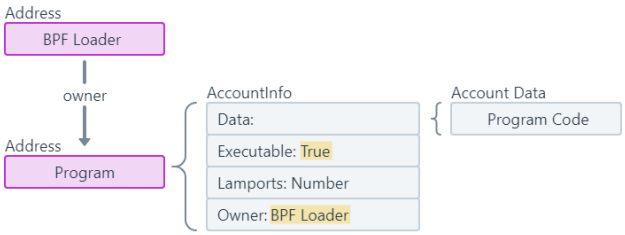


**Ilustración 5. Ejemplo de una cuenta en Solana que es un programa**

Teniendo en cuenta que solo el programa del sistema puede crear cuentas nuevas, crear una cuenta de datos para un programa personalizado requiere dos pasos:

- Invocar el programa del sistema para crear una cuenta, que luego transfiere la propiedad a un programa personalizado
- Invocar el programa personalizado, que ahora es propietario de la cuenta, para inicializar los datos de la cuenta tal y como se definen en el código del programa

***Transacciones.***

En Solana, se envían transacciones para interactuar con la red. Las transacciones incluyen una o más instrucciones, cada una de las cuales representa una operación específica que se procesará. La lógica de ejecución de las instrucciones se almacena en programas implementados en la red Solana, donde cada programa almacena su propio conjunto de instrucciones [7].

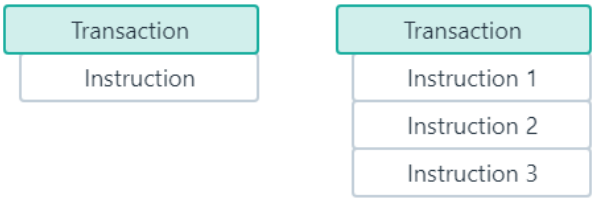


**Ilustración 6. Estructura de una transacción simplificada.**

Una transacción es atómica, lo que significa que se completa por completo y se procesan todas las instrucciones correctamente o falla por completo. Si falla alguna instrucción dentro de la transacción, no se ejecuta ninguna de las instrucciones.

Una transacción de Solana consiste en:

- Firmas: Un arreglo de firmas incluidas en la transacción.
- Mensaje: Lista de instrucciones a procesar atómicamente.

La estructura de un mensaje de transacción consta de:

- Encabezado de mensajes: Especifica el número de cuentas firmantes y de solo lectura.
- Direcciones de cuentas: Un arreglo de direcciones de cuentas requeridas por las instrucciones de la transacción.
- Blockhash reciente: Una especie de marca de tiempo para la transacción. Es utilizado para prevenir duplicaciones y eliminar transacciones abandonadas. La edad máxima del blockhash de una transacción es de 150 bloques (aproximadamente 1 minuto asumiendo que los bloques toman 400ms). Si el blockhash de una transacción es 150 bloques menor que el último blockhash, se considera caducado.
- Instrucciones: Un arreglo de instrucciones a ejecutar.

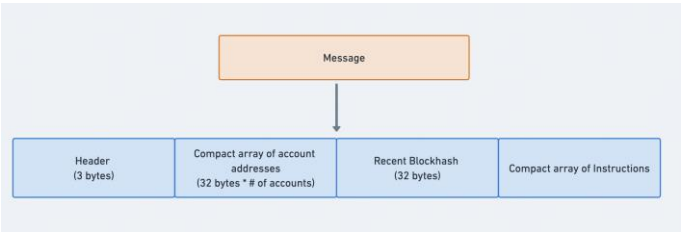


**Ilustración 7. Estructura de un mensaje de la transacción.**

El encabezado de mensajes especifica los privilegios de las cuentas incluidas en el arreglo de direcciones de cuenta de la transacción.

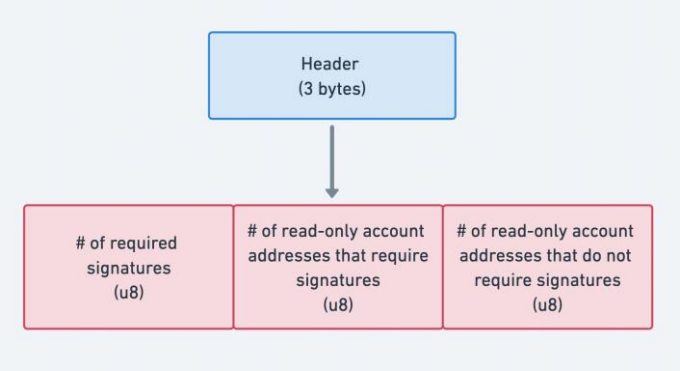


**Ilustración 8. Estructura del encabezado de mensajes.**

Cada instrucción dentro de una transacción tiene tres componentes clave:

- Programa ejecutor: El programa (smart contract) que debe ejecutarse en respuesta a la instrucción. Esto puede ser un programa estándar o uno personalizado en Solana.
- Cuentas requeridas: La lista de cuentas involucradas en la ejecución de la instrucción. Estas cuentas pueden almacenar datos que el programa necesita leer o modificar.
- Datos: Información adicional necesaria para la ejecución, como parámetros específicos que el programa debe recibir para procesar la instrucción.

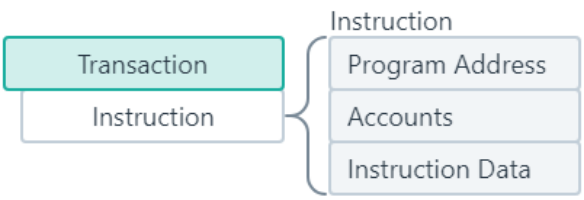


**Ilustración 9. Estructura de una instrucción**

Para cada cuenta requerida por una instrucción, la siguiente información debe ser especificada:

- pubkey: La dirección en la cadena de bloques de una cuenta
- is_signer: Especifica si la cuenta es requerida como un firmante en la transacción
- is_writable: Especifica si los datos de la cuenta serán modificados
-

Esta información se conoce como AccountMeta.

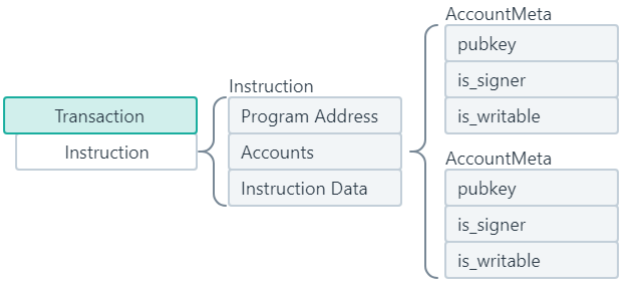


**Ilustración 10. Estructura del AccountMeta**

Asimismo, se debe tener en cuenta que el tamaño máximo de una transacción en Solana es de 1232 bytes. Esto limita la cantidad de instrucciones que se pueden incluir dentro de una única transacción, así como el tamaño de los datos que se pueden enviar con cada instrucción. Este límite asegura que las transacciones sean eficientes en términos de almacenamiento y procesamiento en la red, contribuyendo a la escalabilidad de Solana [18].

Para concluir, se presenta un ejemplo básico de una transacción completa en la red de solana.

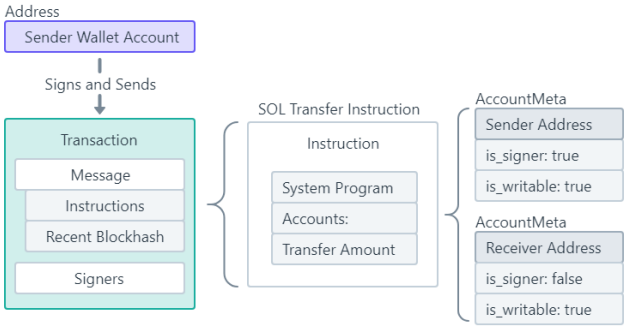


**Ilustración 11. Estructura completa de una transacción.**

Tanto las cuentas de remitente como el destinatario deben ser mutables ***(is_writable***) porque la instrucción modifica el balance de ***lamports*** para ambas cuentas.

Una vez enviada la transacción, se invoca el programa del sistema para procesar la instrucción de transferencia. A continuación, el programa del sistema actualiza los saldos en lamports tanto de la cuenta del remitente como la del destinatario.

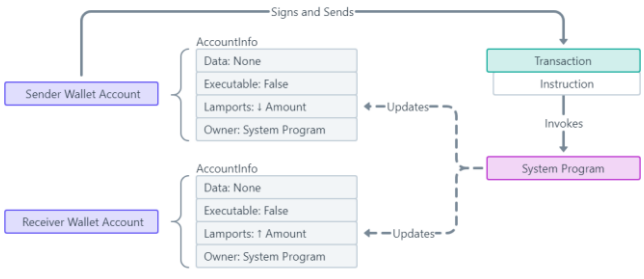


**Ilustración 12. Invocación del programa para procesar la instrucción.**

Comisiones.

La cadena de bloques de Solana tiene diferentes tipos de comisiones y costes que se incurren para utilizar la red [15]. Estos pueden segmentarse tipos específicos:

- Comisiones de Transacción. Estas son las tarifas que se pagan para que los validadores procesen las transacciones e instrucciones dentro de la red. La comisión base es de 5 mil ***lamports*** por firma y se paga al líder validador que procesa la transacción. Además, si se añaden comisiones de prioridad, estas se suman a la base. Además, Las comisiones de transacción ayudan a compensar a los validadores por el uso de recursos de computación, reduciendo el spam en la red y proporcionando estabilidad económica.
- Comisiones de Prioridad: Esta es una tarifa opcional que los usuarios pueden pagar para mejorar el orden de procesamiento de sus transacciones. Esto permite que ciertas transacciones sean procesadas más rápido en comparación con otras.
- Renta: Este es un depósito que se retiene para mantener datos almacenados en la cadena. Es un costo separado de las comisiones de transacción y está diseñado para asegurar que los datos que no se utilizan se eliminen de la red, optimizando el uso del espacio.

Cuando se envía una transacción que incluye varias instrucciones, es el líder validador el que procesa dicha transacción. Si la cuenta que paga la comisión (conocida como "***fee payer***") no tiene suficientes fondos, la transacción falla, y la red no procesa las instrucciones. Sin embargo, se cobra la comisión incluso si la transacción resulta fallida, dado que los validadores ya han utilizado recursos para procesarla.

Por otro lado, las comisiones se deducen del balance de la cuenta que firma la transacción antes de que se procesen las instrucciones. Si alguna instrucción genera un error, se revertirán todos los cambios en la cuenta, excepto la deducción de la comisión.

Las comisiones de transacción se distribuyen de la siguiente manera:

- 50% se quema: Esto implica que se destruye parte de la comisión para estabilizar el valor del token SOL y contribuir a la seguridad de la red.
- 50% al validador: El restante se otorga al validador que produjo el bloque donde se incluyó la transacción.

La comisión completa por una transacción determinada se calcula en función de dos partes principales:

- una comisión base por firma fijada de forma estática, y
- Los recursos computacionales utilizados durante la transacción, medidos en ***unidades de cómputo***.

Dado que cada transacción puede requerir una cantidad diferente de recursos computacionales, a cada una se le asigna un número máximo de unidades de cómputo por transacción como parte del ***presupuesto computacional.*** Este presupuesto especifica detalles sobre las unidades de cómputo e incluye:

- El costo del cómputo asociado a los distintos tipos de operaciones que puede realizar la transacción (unidades de cómputo consumidas por operación)
- El número máximo de unidades de cálculo que puede consumir una transacción (límite de unidades de cómputo),
- Y los límites operativos que debe respetar la transacción (como los límites de tamaño de los datos de la cuenta)

Cada transacción en Solana puede consumir hasta un máximo absoluto de ***1,4 millones*** de unidades de cómputo (CU). Este límite asegura que las transacciones no consuman más recursos de los que la red puede manejar, manteniendo la estabilidad y eficiencia. Además, cada instrucción en una transacción tiene un límite de ***200,000 CU***. Este valor funciona bien en la mayoría de los casos, pero puede ser insuficiente para transacciones más complejas que requieren más recursos computacionales.

**Particularidades**

***Sistema de inflación.***

Al momento del lanzamiento de la red de Solana, se crearon 500 millones de tokens SOL en el bloque génesis, lo que conformó el suministro inicial total [17]. Desde entonces, este suministro ha disminuido debido a dos mecanismos principales:

- Quema de Tarifas por Transacción: Una parte de las tarifas pagadas en las transacciones de la red de Solana, el 50%, se quema, es decir, se elimina permanentemente. Esto reduce el suministro total de SOL, limitando la inflación y favoreciendo una economía sostenible al disminuir la disponibilidad de tokens con el tiempo.

- Evento Planeado de Reducción de Tokens (sistema de inflación): El equipo de Solana también llevó a cabo un evento planificado de reducción de tokens, que contribuyó a disminuir aún más el suministro en circulación. Este tipo de eventos forma parte de una estrategia para controlar la oferta de tokens y mantener su valor a lo largo del tiempo.

El sistema de inflación de Solana está diseñado para apoyar su ecosistema a largo plazo, mientras se ajusta dinámicamente a medida que aumenta el uso de la red y la cantidad de tokens SOL en circulación. Este sistema se estructura a partir de tres parámetros clave: la tasa de inflación inicial, la tasa de desinflación y la tasa de inflación a largo plazo [9].

Al principio, Solana tiene una tasa de inflación relativamente alta, del 8%. Esta tasa significa que, anualmente, la oferta total de SOL en circulación aumenta en un 8%.

Esta emisión inflacionaria está diseñada para recompensar a los participantes que apuestan (stake) sus SOL en la red, ayudando a asegurar la blockchain y validar transacciones. La mayoría de los SOL emitidos a través de la inflación se distribuyen entre los validadores y los delegadores en función de la cantidad de tokens que hayan apostado. Además, la inflación también genera ingresos para los validadores, quienes toman comisiones de los rendimientos generados por el staking de los delegadores [10].

Por otro lado, La tasa de desinflación indica la velocidad a la que la tasa de inflación disminuye con el tiempo. En el caso de Solana, la inflación disminuye un 15% anualmente a partir del 8% inicial. Este enfoque desinflacionario está diseñado para reducir gradualmente la emisión de nuevos tokens, a medida que la red madura y el uso de Solana se incrementa, lo que disminuye la necesidad de una alta inflación para incentivar la seguridad.

A medida que la red madura y la tasa de inflación disminuye, el sistema está diseñado para alcanzar una tasa de inflación a largo plazo del 1.5%. Esta tasa es lo suficientemente baja como para evitar una devaluación significativa del SOL, pero lo suficientemente alta como para seguir incentivando a los validadores y mantener la seguridad de la red.

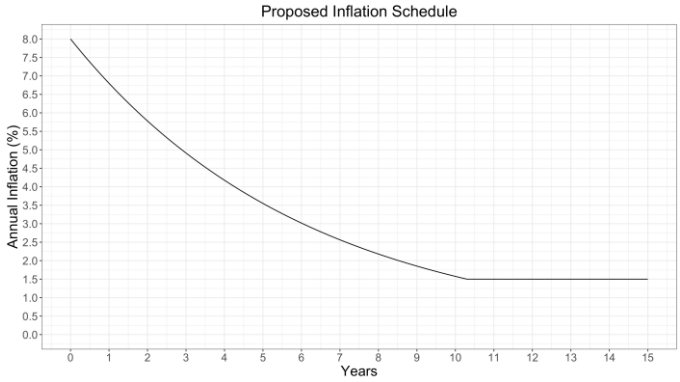


**Ilustración 13. Tasa de inflación anual**

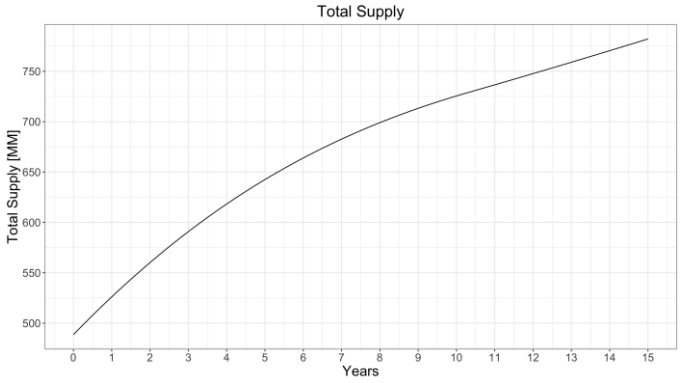


**Ilustración 14. Suministro total de solana Anual**

A medida que aumenta el uso de la red de Solana, se espera que las tarifas por transacción y otras fuentes de ingresos (como alquileres o quema de tarifas) se conviertan en una parte importante de la economía de la red, lo que podría compensar la disminución de las recompensas inflacionarias.

***Extensiones de Token***

Son características integradas para crear tokens avanzados que cumplen con requisitos normativos y de personalización sin necesidad de contratos personalizados. Diseñadas para instituciones, estas extensiones permiten implementar funcionalidades como transferencias confidenciales, tarifas, y tokens con intereses, optimizando seguridad y escalabilidad. Solana simplifica el desarrollo, mejora la personalización y reduce riesgos, tiempo y costos, facilitando la creación de soluciones en áreas como DeFi, gaming y DAOs [22].

Las principales características de las extensiones de token de Solana incluyen:

- Transfer Hooks: Personalizan transacciones, añadiendo condiciones específicas para su ejecución.
- Confidential Transfers: Proporcionan privacidad mediante el cifrado de datos de las transacciones.
- Permanent Delegate: Asigna un delegado para realizar transacciones en nombre del propietario.
- Metadata: Permite añadir información extra al token, como detalles del activo.
- Non-transferable Tokens: Crean tokens que no se pueden transferir, útiles para certificaciones o credenciales.

Las aplicaciones de extensiones de tokens ofrecen una gran versatilidad a desarrolladores y entusiastas por participar en el mercado descentralizado, desde stablecoins hasta memecoins, esta característica de su Blockchain expande los límites de creatividad e innovación. Algunas aplicaciones reales son.

- Paxos USDP: Esta stablecoin utiliza extensiones como Permanent Delegate para cumplir con regulaciones, lo cual permite revocar fondos en caso de actividad maliciosa. Además, incorpora el Confidential Transfer extension, que permite que las transacciones sean confidenciales a través de pruebas de conocimiento cero (Zero-Knowledge Proofs) [13].

- Doland Tremp (TREMP): Inspirada en una sátira política, esta moneda toma el nombre de una versión humorística de Donald Trump. TREMP ha captado la atención con su enfoque paródico y sus memes políticos, lo que lo convierte en un punto de encuentro entre la política y la criptocultura en Solana. Este token se utiliza para crear memes y comentarios políticos, atrayendo a usuarios tanto por su simbolismo como por su potencial para el entretenimiento y la inversión [12].

- Dogwifhat (WIF): Este token ha ganado notoriedad al enfocarse en la "cultura de memes", y presenta un diseño completamente descentralizado, sin un equipo de control central. La comunidad es una parte fundamental del éxito de Dogwifhat, que ha

experimentado un aumento rápido en su valor, impulsado por el entusiasmo y la actividad de sus usuarios [11].

Estos tokens reflejan cómo la blockchain de Solana permite a los desarrolladores y comunidades crear tokens con un enfoque más lúdico y cultural, a menudo con un toque satírico y una fuerte participación de la comunidad.

## Aplicaciones

### *DRIP*

Es una plataforma diseñada para facilitar la creación, distribución y monetización de contenido de creadores mediante NFTs (tokens no fungibles) y tecnología blockchain de Solana. Con DRIP, los usuarios pueden recibir obras de arte digitales gratuitas y semanales mediante airdrops (entregas automáticas de tokens). Una de sus principales características es la eficiencia en el coste de emisión de estos NFTs, gracias a la compresión de datos en Solana, lo que reduce drásticamente el coste de distribución de grandes volúmenes de NFTs, haciéndolo accesible para creadores y sus seguidores [19].

### *Teleport*

Es un sistema de internet para servicios de entrega diseñado para conectar directamente a pasajeros y conductores, eliminando la necesidad de intermediarios como Uber. Este enfoque busca optimizar tanto la experiencia del usuario como la rentabilidad para los conductores [20].

Teleport ofrece múltiples métodos de pago, incluyendo opciones tradicionales como tarjetas de crédito y débito. Además, permite pagos a través del token de Solana, lo que fomenta la adopción de criptomonedas y facilita las transacciones en el ecosistema de Solana.

Así pues, la compañía representa una evolución en la forma en que se gestionan los servicios de transporte y entrega, priorizando la conexión directa entre usuarios y proveedores de servicios, lo que beneficia a ambas partes y mejora la eficiencia del sistema en general.

### *Orca*

Es una plataforma de intercambio descentralizado (DEX) construida sobre la blockchain de Solana, que se centra en proporcionar una experiencia de usuario optimizada y eficiente para el intercambio de activos digitales [21].

Orca permite a los usuarios intercambiar tokens de forma directa sin la necesidad de un intermediario. Esto se realiza a través de contratos inteligentes, lo que garantiza que las transacciones sean seguras y transparentes. Además, gracias a la alta eficiencia de la blockchain de Solana, las tarifas de transacción en Orca son considerablemente más bajas en comparación con otras plataformas, lo que la hace más accesible para los usuarios.

## Implementación de programas

Los programas en Solana son componentes fundamentales que permiten a los desarrolladores implementar la lógica de las aplicaciones descentralizadas (dApps) en la cadena de bloques de Solana.

### *Lenguajes de programación y Frameworks*

- Rust: Los programas de Solana están predominantemente escritos en Rust, un lenguaje de programación de sistemas conocido por su seguridad y rendimiento. Rust permite a los desarrolladores escribir código eficiente y seguro, lo que es crucial en un entorno blockchain.
- Anchor: es un ***framework*** diseñado específicamente para el desarrollo de programas en Solana. Proporciona una manera más rápida y sencilla de escribir programas al utilizar macros de Rust que reducen significativamente el código repetitivo (boilerplate).
- Rust nativo: Este enfoque implica escribir programas en Rust sin utilizar un marco como Anchor. Si bien proporciona más flexibilidad y control sobre el código, también conlleva una mayor complejidad y responsabilidad por parte del desarrollador.

### *Verificación de programas.*

La comunidad de desarrolladores de Solana ha introducido herramientas para apoyar las compilaciones verificables, facilitando a los desarrolladores y usuarios verificar que los programas en la cadena reflejan con precisión su código fuente compartido públicamente. Para esto, los usuarios pueden buscar un programa por su dirección en el explorador ***SolanaFM*** y acceder a la pestaña de Verificación para comprobar rápidamente si un programa está verificado [16].

### *Compilador BFP*

Solana utiliza la infraestructura del compilador LLVM para compilar programas en archivos ejecutables y enlazables (ELF). Estos archivos contienen una versión modificada del bytecode Berkeley Packet Filter (eBPF), conocida como Solana Bytecode Format (sBPF).

El uso de LLVM permite a Solana soportar potencialmente cualquier lenguaje de programación que pueda compilarse al backend BPF de LLVM, mejorando la flexibilidad de Solana como plataforma de desarrollo.

### *Desarrollo de programas*

El desarrollo en Solana se organiza en dos áreas principales: la construcción de programas en la cadena de bloques y el desarrollo de clientes (dApps). La primera área se refiere a la creación de contratos inteligentes utilizando lenguajes de programación como Rust, C y C++. En contraste, el desarrollo de clientes implica crear software (dApps) que se comunica con estos programas desplegados en la cadena de bloques. Las

dApps envían transacciones a la cadena para ejecutar acciones específicas definidas por los programas a través de la API de JSON RPC, lo que permite una interacción fluida y accesible para todos los usuarios.

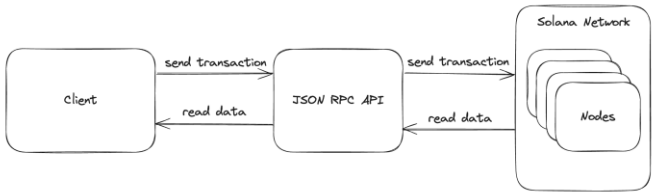


**Ilustración 15. Comunicación cliente y programa.**

***Creación de un programa con Anchor***

Para crear un programa en la Testnet de solana, se puede hacer uso de un entorno de desarrollo que provee la Blockchain aquí [25].

Una vez en el ambiente, lo siguiente será crear un nuevo proyecto utilizando el Framework Anchor.

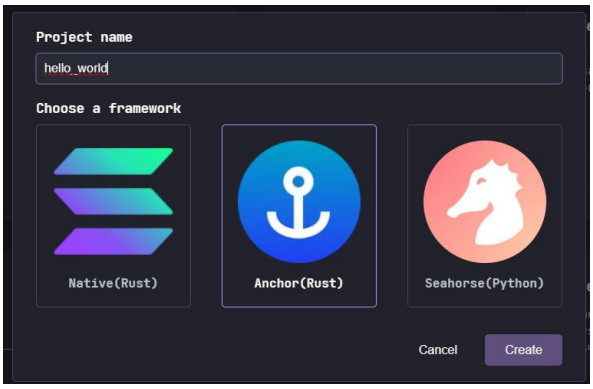


**Ilustración 16. Creación de un programa con Anchor**

Luego de finalizar la construcción, el programa base tendrá un código por defecto como el siguiente.

```rust
use anchor_lang::prelude::*;

// This is your program's public key and it will update
// automatically when you build the project.
declare_id!("11111111111111111111111111111111");

#[program]
mod hello_anchor {
    use super::*;
    pub fn initialize(ctx: Context<Initialize>, data: u64) -> Result<()> {
        ctx.accounts.new_account.data = data;
        msg!("Changed data to: {}!", data); // Message will show up in the tx logs
        Ok(())
    }
}

0 implementations
#[derive(Accounts)]
pub struct Initialize<'info> {
    // We must specify the space in order to initialize an account.
    // First 8 bytes are default account discriminator,
    // next 8 bytes come from NewAccount.data being type u64.
    // (u64 = 64 bits unsigned integer = 8 bytes)
    #[account(init, payer = signer, space = 8 + 8)]
    pub new_account: Account<'info, NewAccount>,
    #[account(mut)]
    pub signer: Signer<'info>,
    pub system_program: Program<'info, System>,
}
```

**Ilustración 17. Programa base en Solana**

Con el fin de comprender mejor la lógica de lo que se está realizando, se detallarán algunas sentencias clave [23].

La línea,
***declare_id!("11111111111111111111111111111111");*** define el ID público del programa, que es un identificador único. Al compilar el programa, Anchor asignará un nuevo ID automáticamente.

El módulo ***hello_anchor*** define el programa en sí:

- La función initialize recibe un contexto (ctx) y un dato (data), que es de tipo u64 (un número entero sin signo de 64 bits).
- initialize guarda el valor de data en una cuenta llamada new_account, que se obtiene del contexto (ctx.accounts.new_account.data = data).
- La función msg! envía un mensaje de log que puede verse en los registros de la transacción (tx logs).

La estructura Initialize define los participantes necesarios (las cuentas) para ejecutar initialize:

- new_account: Es la cuenta que el programa creará e inicializará. Está definida con el atributo #[account(init, payer = signer, space = 8 + 8)], que indica:
  - init: La cuenta se creará en la blockchain cuando se llame a initialize.
  - payer = signer: La cuenta signer pagará los costos de creación de new_account.
  - space = 8 + 8: Define el espacio que ocupa new_account. Los primeros 8 bytes son para el discriminador de cuenta (identificador que Anchor utiliza para verificar el tipo de cuenta), y los siguientes 8 bytes son para el u64 almacenado en new_account.
- signer: Esta cuenta es quien firma y paga la creación de new_account, y está definida con #[account(mut)], lo que significa que su saldo puede cambiar (por ejemplo, al pagar la transacción).
- system_program: Es el programa de sistema de Solana, necesario para inicializar cuentas en la blockchain.

Teniendo claro los elementos esenciales del código, ahora se puede generar el Build (compilación) y desplegar el programa haciendo uso de las herramientas proporcionadas en el playground.

Al compilar el programa, ahora Anchor ha asignado un ID único a él, que en este caso corresponde al siguiente.

```rust
// This is your program's public key and it will update
// automatically when you build the project.
declare_id!("76b3jdFMpuWnjuCZUphU27DVuDVVdL3FBHPhLTE3H1Vy");
```

**Ilustración 18. Id único del programa**

Por último, para desplegar el contrato en la Testnet, primero se debe solicitar airdops de Sol a la wallet asignada en el playground aquí [24].

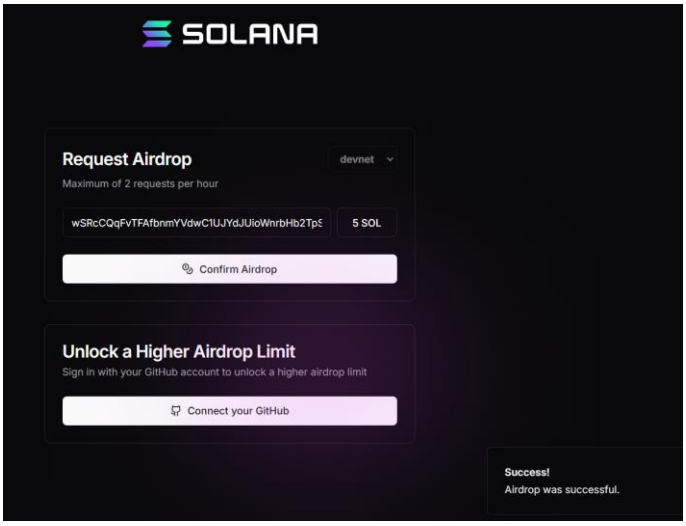


**Ilustración 19. Solicitud de Airdrop en Sol**

Ahora, la wallet contiene un balance suficiente para desplegar el programa.


**Ilustración 20. Balance de la wallet**

Por último, en la opción IDL, se puede exportar el código del contrato desplegado para que posteriormente sea usado desde el lado del cliente. El archivo de exportación resultante será en formato Json.

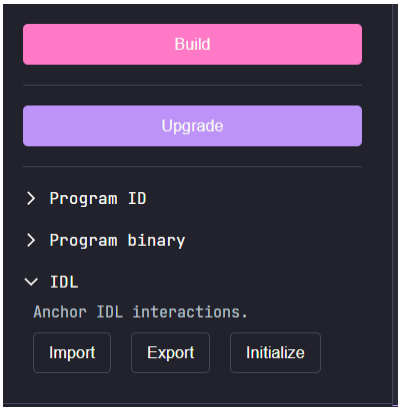


**Ilustración 21. Exportar el programa a través del IDL**

## Discusión.

Solana presenta un avance significativo en el espacio de las blockchain al ofrecer una solución innovadora para la gestión del tiempo y la eficiencia en la validación de transacciones. Su implementación de Proof of History (PoH) no solo resuelve el problema de la secuencia temporal, sino que también permite una sincronización confiable y rápida en la red. Esto, combinado con el mecanismo de proof of stake, le da a Solana una ventaja en términos de sostenibilidad energética, costos bajos y alta velocidad, características que son altamente deseables en el ecosistema de blockchain y han permitido a Solana procesar miles de transacciones por segundo.

Entre los puntos positivos de Solana, se destacan su estructura eficiente y su capacidad para soportar una gran cantidad de transacciones a costos mínimos, lo cual la convierte en una excelente opción para aplicaciones descentralizadas, DeFi y NFTs. Las innovaciones en su sistema de cuentas y transacciones facilitan la gestión y extensión de tokens, mientras que su enfoque en escalabilidad y sostenibilidad abre la puerta para una adopción masiva. Además, Solana ofrece un ambiente de desarrollo robusto a través de su playground y el framework Anchor, que simplifica la creación de programas en la red y ha permitido el surgimiento de aplicaciones destacadas como Orca y Drip.

No obstante, Solana aún enfrenta desafíos importantes.Su sistema de inflación y el modelo de validación, aunque innovadores, necesitan una supervisión continua para asegurar la estabilidad económica de la red y mantener la participación de los validadores.

En conclusión, Solana representa una de las propuestas más innovadoras en el mundo de blockchain, combinando eficiencia, velocidad y bajos costos. Con mejoras en su infraestructura y estabilidad, Solana tiene el potencial de consolidarse como una de las principales plataformas para la próxima generación de aplicaciones descentralizadas.

## Referencias.

[1] Solana, "Digital Assets," Solana.com. [Online]. Available: https://solana.com/es/solutions/digital-assets.

[2] A. Yakovenko, "Solana: A new architecture for a high performance blockchain," Solana.com.

[3] Solana, "Blockchain News," Solana.com. [Online]. Available: https://solana.com/news/tag/blockchain.

[4] Cointelegraph, “¿Qué es Solana y cómo funciona?”, Cointelegraph, 2024. [Online]. Available: https://es.cointelegraph.com/news/what-is-solana-and-how-does-it-work.

[5] Solana, “Writing Programs: Rust”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/programs/lang-rust.

[6] Solana, “Accounts”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/core/accounts.

[7] Solana, “Transactions”, Solana Documentation, 2024. [Online]. Available: https://solana.com/docs/core/transactions.

[8] J. Nwosu, “Solana Explained in 10 Minutes”, YouTube, 2021. [Online]. Available: https://www.youtube.com/watch?v=CJOEkJNz6dU.

[9] Solana, “Inflation Schedule”, *Solana Documentation*, 2024. [Online]. Available: https://solana.com/docs/economics/inflation/inflation-schedule.

[10] Solana, “Terminology,” Solana Documentation, 2024. [Online]. Available: https://solana.com/es/docs/economics/inflation/terminology

[11] Block Insider, “Top 5 Memecoins on Solana with the Strongest Community,” Block Insider. [Online]. Available: https://blockinsider.com/crypto/top-5-memecoins-on-solana-with-the-strongest-community

[12] Trust Wallet, “Doland Tremp: Solana Memecoin Explained,” Trust Wallet Blog, 2024. [Online]. Available: https://trustwallet.com/es/blog/doland-tremp-solana-memecoin-explained

[13] QuickNode, “SPL Tokens: Token 2022 Overview,” QuickNode. [Online]. Available: https://www.quicknode.com/guides/solana-development/spl-tokens/token-2022/overview

[14] D. J. Bernstein, “Ed25519,” Ed25519 Cryptography, 2024. [Online]. Available: https://ed25519.cr.yp.to/.

[15] Solana, “Fees,” Solana Documentation, 2024. [Online]. Available: https://solana.com/es/docs/core/fees.

[16] Solana, “Introduction for Developers,” Solana Documentation, 2024. [Online]. Available: https://solana.com/es/docs/intro/dev.

[17] Solana Explorer, “Current Supply of SOL Tokens,” Solana Explorer. [Online]. Available: https://explorer.solana.com/supply.

[18] Solana, “The Solana programming model,” YouTube, 2023. [Online]. Available: https://www.youtube.com/watch?v=pRYs49MqapI.

[19] Solana, “Only posible on Solana, Drip” YouTube, 2024. [Online]. Available: https://www.youtube.com/watch?v=XAND6G6dW2c.

[20] Solana, “Only posible on Solana, Teleport” YouTube, 2024. [Online]. Available: https://www.youtube.com/watch?v=gknWJoqyO2M.

[21] Solana, “Only posible on Solana, Orca” YouTube, 2023. [Online]. Available: https://www.youtube.com/watch?v=FXUsPjDvfcs.

[22] Trust Wallet, “Solana Token Extensions Explained,” Trust Wallet Blog, 2024. [Online]. Available: https://trustwallet.com/es/blog/solana-token-extensions-explained.

[23] Anchor, "Anchor Documentation," Anchor-lang.com. [Online]. Available: https://www.anchor-lang.com/.

[24] Solana, "Solana Faucet," Faucet.solana.com. [Online]. Available: https://faucet.solana.com/.

[25] SolPG, "Beta SolPG," SolPG.io. [Online]. Available: https://beta.solpg.io/.