

UTILIDADES BLOCKCHAIN

Usando la herramienta “Hyperledger Sawtooth”



2023

Ing. Marco Aurelio Guado Zavaleta
Master en Desarrollo Software Corporativas
<https://www.linkedin.com/in/marcoguado/>

INTRODUCCION

Blockchain es una tecnología de registro distribuido que permite a múltiples partes tener acceso a una base de datos compartida, sin necesidad de una autoridad central que la controle. Esta tecnología fue popularizada por la criptomoneda Bitcoin, pero hoy en día se utiliza en una amplia gama de aplicaciones, desde sistemas de votación hasta cadenas de suministro.

Hyperledger Sawtooth es una plataforma de Blockchain de código abierto desarrollada por la Fundación Linux. Fue diseñada para ser modular y escalable, lo que la hace adecuada para aplicaciones empresariales complejas. Sawtooth utiliza una variedad de algoritmos de consensos para validar transacciones y crear nuevos bloques en la cadena de bloques. También incluye características avanzadas, como la capacidad de implementar contratos inteligentes en varios lenguajes de programación y la gestión de permisos granulares para los usuarios de la red. En resumen, Hyperledger Sawtooth es una plataforma de Blockchain potente y flexible que puede adaptarse a las necesidades específicas de una empresa.

Este documento describirá lo que se necesita para implementar un 'Smart Contract' como:

- La instalación de un Nodo Blockchain.
- Desarrollar un 'Smart Contract' (contrato inteligente).
- Utilizar el 'Smart Contract' por intermedio de una API (Interfaz de programación de aplicaciones).

INDICE

1. Definición
2. Base de datos descentralizada
3. Base de datos centralizada
4. Registrar datos en Blockchain
5. Blockchain servicios
6. Plataformas Blockchain
7. Smart Contract
8. Configurar entorno desarrollo
9. Desarrollar el Smart Contract
10. Utilizar el Smart Contract.

1. Definición

Blockchain es una tecnología de registro distribuido que se utiliza para almacenar y verificar transacciones en una red distribuida. Se basa en una base de datos descentralizada y segura, en la que los datos se registran en bloques interconectados y cifrados para garantizar su integridad y seguridad.

2. Base de datos descentralizada

Una base de datos descentralizada almacena y distribuye la información (datos) en múltiples nodos o dispositivos, en lugar de estar centralizada en un solo servidor o ubicación. En este tipo de base de datos, cada nodo tiene una copia completa de los datos y las actualizaciones se propagan a través de la red para mantener todas las copias sincronizadas.

La ventaja de una base de datos descentralizada es que ofrece una mayor resistencia a la falla y una mayor seguridad, ya que no hay un único punto de falla o ataque. Además, una base de datos descentralizada puede permitir la colaboración y la participación de múltiples usuarios sin necesidad de una autoridad central, lo que puede ser útil en aplicaciones donde se requiere transparencia y confianza, como las aplicaciones de cadena de bloques (Blockchain).

2.1. Características

Las características de una base de datos descentralizada son:

- **Distribución:** La información se almacena y se distribuye en múltiples nodos o dispositivos, lo que significa que no hay un único punto de falla o ataque.
- **Redundancia:** Cada nodo tiene una copia completa de los datos, lo que significa que, en caso de que un nodo falle, la información todavía estará disponible en otros nodos.
- **Seguridad:** Las medidas de seguridad y privacidad se pueden implementar en cada nodo individualmente para proteger la información.
- **Transparencia:** La información se puede acceder y actualizar de manera transparente por todos los usuarios que tienen acceso a la base de datos.
- **Participación:** Los usuarios pueden participar en la base de datos sin necesidad de una autoridad central, lo que puede ser útil en aplicaciones donde se requiere confianza y transparencia.
- **Sincronización:** Las actualizaciones se propagan a través de la red para mantener todas las copias de los datos sincronizadas.
- **Escalabilidad:** La base de datos descentralizada puede escalar de manera eficiente a medida que se agregan más nodos a la red.

En resumen, una base de datos descentralizada proporciona una mayor resistencia a la falla y una mayor seguridad, y permite la colaboración y participación de múltiples usuarios sin la necesidad de una autoridad central.

3. Base de datos (centralizada)

Una base de datos centralizada almacena la información en un solo servidor o ubicación centralizada. En este tipo de base de datos, los usuarios acceden a los datos a través de una red, y todas las actualizaciones y modificaciones se realizan en el servidor central.

La ventaja de una base de datos centralizada es que puede ser más fácil de administrar y mantener, ya que todos los datos están en un solo lugar y se pueden implementar medidas de seguridad y privacidad en un solo punto. Sin

embargo, también presenta algunas desventajas, como la falta de redundancia y la posibilidad de un único punto de falla, lo que puede aumentar el riesgo de pérdida de datos y fallas en el sistema.

Además, en una base de datos centralizada, es posible que se requiera una autoridad central para administrar y controlar el acceso a los datos, lo que puede limitar la transparencia y la participación de múltiples usuarios.

3.1. Características

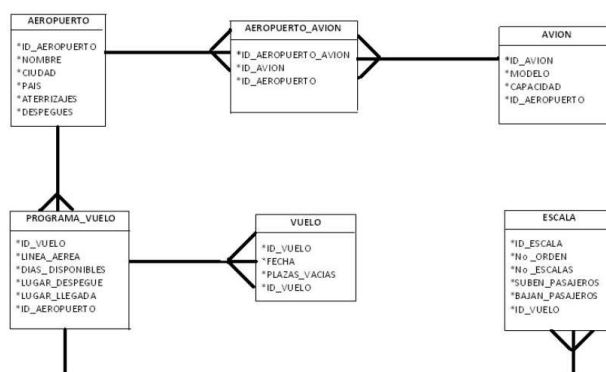
Las características de una base de datos centralizada son:

- Centralización: La información se almacena en un solo servidor o ubicación centralizada.
- Administración: La administración y el mantenimiento de la base de datos se realizan en un solo lugar, lo que facilita su gestión.
- Control de acceso: El control de acceso a la base de datos se realiza en el servidor central, lo que permite implementar medidas de seguridad y privacidad de manera efectiva.
- Escalabilidad limitada: La base de datos centralizada tiene una capacidad limitada de escalar a medida que se agregan más datos y usuarios.
- Vulnerabilidad a fallas: Un único punto de falla puede provocar la pérdida de toda la información almacenada en la base de datos.
- Dependencia de la conexión de red: La conexión de red es esencial para acceder a la base de datos, lo que puede limitar la disponibilidad de la información si hay problemas de conexión.
- Dependencia de la autoridad central: La base de datos centralizada depende de una autoridad central para administrar y controlar el acceso a los datos, lo que puede limitar la transparencia y la participación de los usuarios.

En resumen, una base de datos centralizada ofrece una administración y un control de acceso centralizado, pero presenta limitaciones en términos de escalabilidad, vulnerabilidad a fallas y dependencia de la conexión de red y la autoridad central.

Ejemplos de Bases de datos centralizadas y descentralizadas

Las Bases de datos (centralizadas) almacenan la información en estructuras conocidas como filas columnas (algo similar como si fueran archivos Excel) estas estructuras se llaman tablas y puede haber N tablas en una Base de datos que representen el modelo de datos de un negocio. Ejemplo, en la siguiente imagen representamos el modelo de datos de un aeropuerto (visto de modo sencillo)

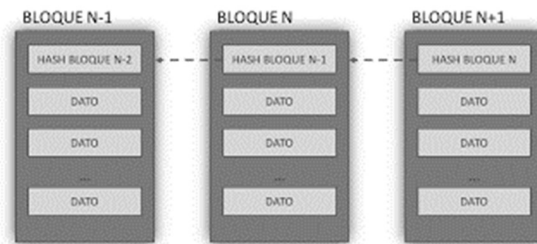


Copyright <http://basenatos.blogspot.com/2009/02/ejercicio-4-sistema-de-vuelos.html>

Bajo este modelo podemos consultar en la Base de datos; que tipos de vuelos tienen escala, que vuelo usan que tipo de avión, que aeropuertos tiene vuelos programados por la noche etc.

En resumen, las Bases de datos (centralizadas) almacenan los datos bajo un esquema que agrupa N tablas en este caso; las Tablas son; Aeropuerto, Vuelo, Scala etc.

Blockchain al ser una Base de datos distribuida almacena la información de manera secuencial como si fuera un tren, pero solo almacena información extremadamente útil y que se pueda compartir.



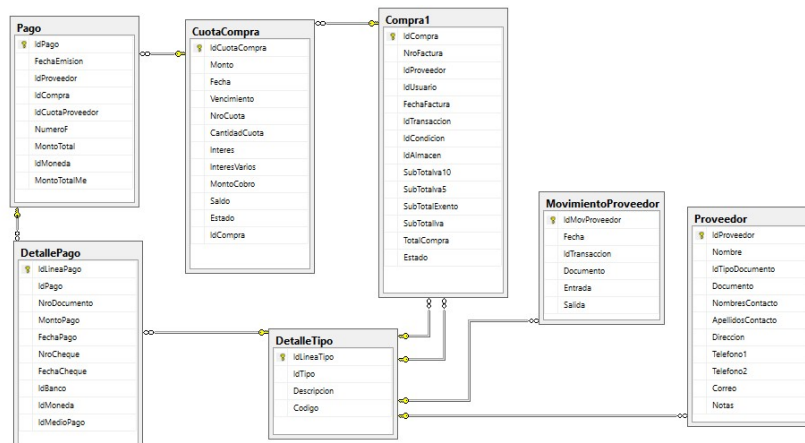
Copyright <https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-ii-conceptos-basicos-proteccion-de-datos>

Que datos podríamos almacenar siguiendo el ejemplo del Aeropuerto, podría ser:

DATO = “Vuelo KL 744, AMS – LIM, pasj 234, combustible 20 tn, Aeropuerto Jorge Chavez”

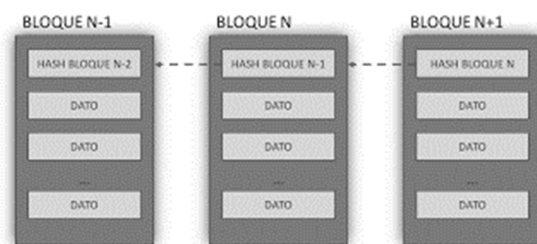
Blockchain es una base de datos distribuida que almacena los datos de forma diferente, no es en estructura sino en secuencial.

Veamos otro ejemplo de una Base de datos (centralizada) sistema de pagos, con varias tablas:



Copyright <https://social.msdn.microsoft.com/Forums/sqlserver/es-ES/e7305f9d-9e2a-4c96-83a8-fd8d5de8920e/diagrama-entidad-relacion?forum=vcses>

Pero en un sistema de almacenamiento Blockchain de PAGOS, solo sería útil registrar el pago.



Copyright <https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-ii-conceptos-basicos-proteccion-de-datos>

DATO = “PAGO, MontoTotal=123,NoCuota=2,Fecha=12/2/2023”

Otra vez, las cadenas Blockchain solo almacenan información que es importante para el negocio, en este caso se almacena el pago en fecha y monto y que la información no será manipulada y además tenemos la seguridad que lo podemos compartir, por ejemplo, para temas de auditorías.

Otro ejemplo de uso de servicios de Blockchain es en la venta de entradas ya que el ticket no puede ser copiado ni alterado, el dato a guardar sería algo así:

DATO =" PART-PERUvsALEMANIA-234N565M4-02/03/2023-0"

Si la entrada es consumida, la información **cambiaría** y se registraría de esta manera:

DATO =" PART-PERUvsALEMANIA-234N565M4-02/03/2023-1"

Con esto logramos inutilizar la entrada.

"Todos estos ejemplos comentados, están bajo el servicio Blockchain de tipo "Smart Contract" ya que necesitan ejecutar ciertas reglas antes de guardar la información"

Nota. - en la práctica un "Smart Contract" son líneas de programación que implementan una funcionalidad, por ejemplo, para poder tener el registro DATO podemos programar acopiando la información de las tablas; evento, usuario, pago y con esos datos creamos un solo registro (DATO) que vamos a guardar en la Blockchain.

4. Registrar datos en Blockchain

Para registrar los datos en Blockchain el usuario debe hacerlo por medio de un certificado privado y puede delegar su certificado público a otros usuarios para que puedan ver la información, esta es una de las utilidades más importantes que aseguran que la información nunca será manipulada.

En las Bases de datos (centralizadas) los usuarios tienen permisos para escribir y leer, en cambio en Blockchain se escribe usando el certificado privado y se lee utilizando el certificado público.

Todo usuario que interactúa con la Blockchain debe hacerlo usando sus certificados tanto privado como públicos. Sin los certificados no se podrá acceder a la información.

4.1. Certificados privados y públicos

Los certificados privados y públicos son componentes clave de la infraestructura de Blockchain y se utilizan para autenticar la identidad de un usuario o sistema para poder interactuar en la Blockchain. Los certificados por lo general son ficheros físicos que tienen almacenado en forma encriptada los datos del usuario, estos certificados son emitidos por los que gestionan la infraestructura Blockchain en la que interactúan.

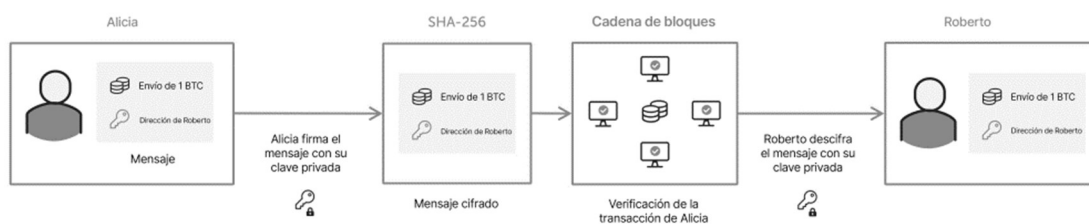
La siguiente imagen grafica la utilidad de los certificados, si bien en la gráfica representa criptomonedas, el procedimiento es el mismo para un dato.

1. Envío un dato, por ejemplo, el de la compra de entrada:

Mensaje => DATO =" PART-PERUvsALEMANIA-234N565M4-02/03/2023"

Y para validar que ha sido el usuario Roberto, lo firma usando su certificado privado (clave privada)

2. El mensaje firmado se encuentra cifrado y se pasa a la cadena de Bloques.
3. El usuario Roberto y solo él usando su certificado público (clave pública) puede leer el contenido de la información.



Copyright <https://www.ledger.com/es/academy/que-son-las-claves-publicas-y-privadas>

Nota. - ningún hacker ha quebrado la seguridad de Blockchain, en todas las noticias a fecha que se han publicado sobre el robo de Blockchain, los hackers se han apoderado de los certificados (siendo este su talón de Aquiles) Blockchain.

Pero para corregir este punto de seguridad se siguen varias estrategias, una es delegar a cada usuario la custodia de sus certificados o dos tener un solo usuario que valide todas las transacciones.

5. Blockchain servicios

La tecnología Blockchain puede ser útil para implementar 3 servicios:

- Las criptomonedas.
- Los contratos inteligentes conocidos como: “Smart Contract”
- Los tokens NFS que sirven para identificar la autenticidad de un archivo digital.

Pregunta: ¿Bitcoin es Blockchain? Bitcoin es un servicio que utiliza la tecnología Blockchain y está en la categoría de criptomonedas.

Aparte de utilizar Blockchain como generador de criptomonedas, existen otros servicios como los “Smart Contract” que es lo que se utiliza en la mayoría de sistemas Blockchain cuando procesan datos, por último, tenemos los NFTs que conjuntamente con los “Smart Contract” sirven para asegurar la autenticidad y originalidad de algún contenido digital. Si diseñamos un logo y lo distribuimos en Internet después como podemos reclamar derechos de autor, esto se logra uniendo NFT’s + Smart Contract.

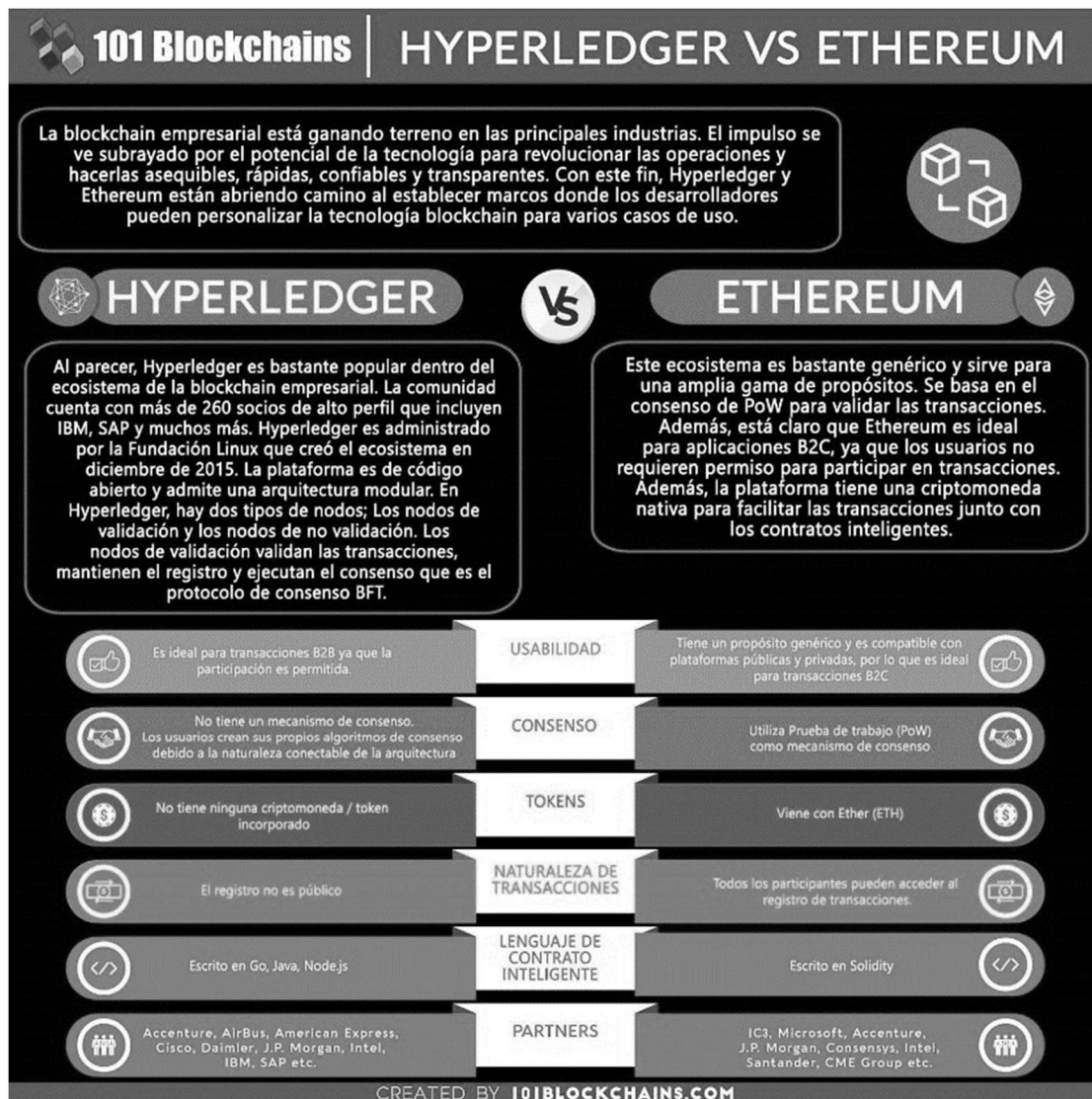
Otro ejemplo es en temas de auditoria donde podemos guardar documentos asegurando su autenticidad y asegurar que hasta tal fecha no fueron manipulados. En criminalística utilizar tecnología Blockchain es muy útil para preservar los datos de una prueba.

En resumen: La tecnología Blockchain es muy útil cuando manipula datos y queremos asegurar que esos valores no serán alterados y tenemos la confianza de poder compartir la información a terceros.

Autenticidad y confianza son los grandes valores de la tecnología Blockchain

5.1. Tipos de Blockchain

No todas las plataformas Blockchain llevan asociadas una moneda. En este caso Hyperledger Sawtooth para cifrar datos no necesita de una moneda, en cambio Ethereum al tener una moneda es parte de su negocio por cada transacción (cifrar) utiliza una moneda. En la siguiente imagen podemos ver un cuadro comparativo entre Hyperledger y Ethereum.



Copyright - <https://101blockchains.com/es/hyperledger-or-ethereum/>

6. Plataformas Blockchain

Para programar utilizando tecnología Blockchain existe muchas plataformas, en el siguiente **grafico** se representan algunas, pero existen muchas más.

Summary of Features of top 5 Blockchain Platforms for Enterprises

	Ethereum	Hyperledger Fabric	R3 Corda	Ripple	Quorum
Industry-focus	Cross-industry	Cross-industry	Financial Services	Financial Services	Cross-industry
Governance	Ethereum developers	Linux Foundation	R3 Consortium	Ripple Labs	Ethereum developers & JP Morgan Chase
Ledger type	Permissionless	Permissioned	Permissioned	Permissioned	Permissioned
Cryptocurrency	Ether (ETH)	None	None	Ripple (XRP)	None
% providers with experience ¹	93%	93%	60%	33%	27%
% share of engagements ²	52%	12%	13%	4%	10%
Coin Market Cap ³	\$91.5 B (18%)	Not applicable	Not Applicable	\$43.9 B (9%)	Not Applicable
Consensus algorithm	Proof of Work (PoW)	Pluggable framework	Pluggable framework	Probabilistic voting	Majority voting
Smart contract functionality	Yes	Yes	Yes	No	Yes

¹ Based on responses from 15 leading blockchain service providers
² Based on a random sample of set of 50 enterprise blockchain engagements across multiple industries
³ Coinmarketcap.com as of Feb 20, 2018, 6:20 PM UTC

Source: HFS Research, 2018

© HFS Research 2018

Copyright <http://techdatasmex.blogspot.com/2018/03/las-5-mejores-plataformas-de-blockchain.html>

Como verán en la imagen anterior solo dos herramientas tiene criptomonedas Ethereum y Ripple.

Nota. – no hay necesidad de utilizar una moneda para cifrar datos en la Blockchain de Hyperledger Sawtooth, si nuestro negocio a parte de cifrar datos queremos generar una moneda entonces podemos usar Ethereum.

Podemos comparar la imagen anterior con el siguiente comentario:” si queremos programar páginas web, cuantos lenguajes de programación existen”. Muchos que se representan en esta imagen:

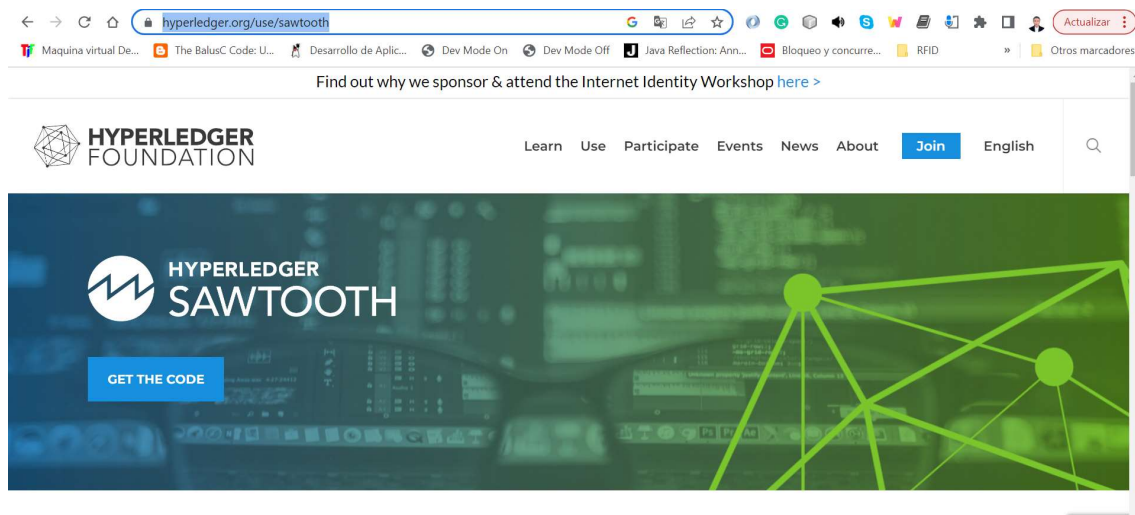


Copyright <https://codigoonclick.com/mejores-lenguajes-programacion-para-2018/>

¡Para programar una página web, debo aprender todos esos lenguajes de programación, claro que no! Basta con elegir un lenguaje.

Lo mismo pasa con Blockchain, para esta guía vamos elegir la plataforma Hyperledger Sawtooth.

Web Hyperledger: <https://www.hyperledger.org/use/sawtooth>



6.1. Hyperledger Sawtooth

Hyperledger Sawtooth es una plataforma de tecnología Blockchain de código abierto desarrollada por la Fundación Linux como parte del proyecto Hyperledger. Sawtooth está diseñado para proporcionar una plataforma empresarial escalable y flexible para aplicaciones Blockchain.

La plataforma se basa en una arquitectura modular y permite la implementación de aplicaciones descentralizadas con una amplia gama de requisitos de rendimiento y privacidad. Sawtooth utiliza varios algoritmos de consenso, además es escalable y consume menos energía que algunos de los otros algoritmos de consenso.

Sawtooth ofrece una serie de características y herramientas que permiten la creación y el despliegue de aplicaciones Blockchain personalizadas, incluyendo un lenguaje de contrato inteligente flexible, una interfaz de línea de comandos para la administración de nodos, un panel de control web para la supervisión y gestión de redes, y una API RESTful para integrar aplicaciones Blockchain con otras aplicaciones empresariales.

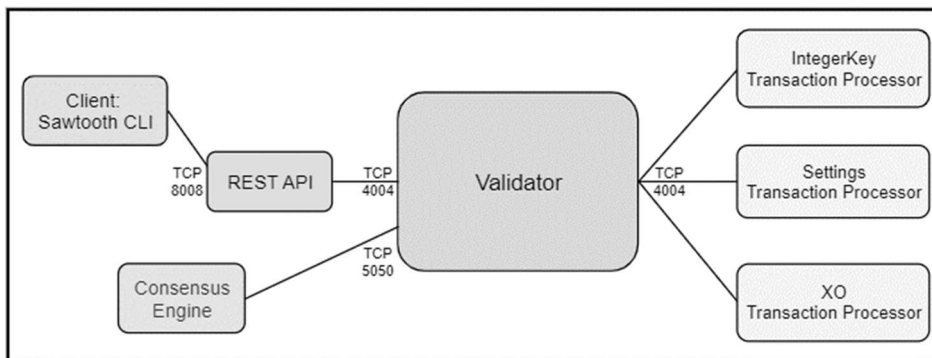
6.2. Características Hyperledger Sawtooth

Hyperledger Sawtooth es una plataforma de Blockchain de código abierto que está diseñada para ser modular, escalable y segura. Aquí te presento algunas de sus características más destacadas:

- **Arquitectura modular:** Sawtooth utiliza una arquitectura modular, lo que significa que las diferentes capas del protocolo de Blockchain, como el consenso, la validación de transacciones y el almacenamiento de datos, se pueden reemplazar y personalizar según las necesidades específicas de la aplicación.
- **Consenso dinámico:** Sawtooth permite el uso de múltiples algoritmos de consenso, lo que permite a los desarrolladores elegir el algoritmo que mejor se adapte a las necesidades de su aplicación. Además, Sawtooth utiliza un modelo de consenso basado en la transacción, lo que significa que las transacciones se agrupan en bloques y se someten a un proceso de validación antes de ser aceptadas en la Blockchain.
- **Modelo de datos flexible:** Sawtooth utiliza un modelo de datos flexible que permite la creación de diferentes tipos de registros, como contratos inteligentes y activos digitales, lo que hace que sea más fácil para los desarrolladores crear aplicaciones específicas.
- **Seguridad:** Sawtooth utiliza técnicas de seguridad avanzadas como el cifrado de transacciones y la autenticación de usuarios para proteger la integridad de la Blockchain.
- **Escalabilidad:** Sawtooth está diseñado para escalar horizontalmente, lo que significa que se puede agregar capacidad a la red simplemente agregando más nodos. Además, Sawtooth utiliza un modelo de partición de estado que permite la paralelización de transacciones, lo que permite que la red maneje un mayor volumen de transacciones.

En resumen, Hyperledger Sawtooth es una plataforma de Blockchain flexible y escalable que permite a los desarrolladores personalizar la arquitectura de Blockchain y elegir el algoritmo de consenso que mejor se adapte a las necesidades de su aplicación. Además, Sawtooth utiliza técnicas de seguridad avanzadas y está diseñado para escalar horizontalmente para manejar un mayor volumen de transacciones.

Arquitectura de un Nodo Blockchain Hyperledger Sawtooth



Copyright https://sawtooth.hyperledger.org/docs/1.2/app_developers_guide/installing_sawtooth.html

6.3. ¿Qué es un nodo?

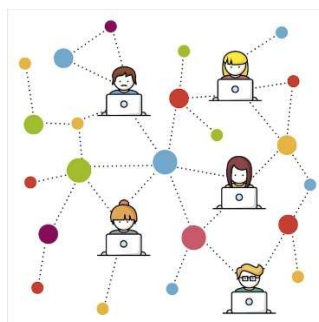
Un nodo Blockchain es un componente esencial de una red Blockchain descentralizada que ayuda a mantener la integridad y seguridad de la cadena de bloques. Un nodo puede ser definido como cualquier dispositivo conectado a la red Blockchain que participe en el proceso de validación de transacciones y confirmación de nuevos bloques en la cadena.

Cada nodo en la red tiene una copia completa de la cadena de bloques, lo que significa que contiene todos los datos y registros de transacciones desde la creación de la cadena. Los nodos trabajan en conjunto para validar transacciones y confirmar nuevos bloques, lo que significa que, si uno de los nodos en la red detecta una transacción inválida o sospechosa, los demás nodos pueden verificarla y determinar si se debe aceptar o rechazar.

Los nodos Blockchain pueden ser divididos en diferentes categorías, según su función. Por ejemplo, los nodos completos (full nodes) tienen una copia completa de la cadena de bloques y participan en la validación de transacciones y la creación de nuevos bloques, mientras que los nodos ligeros (light nodes) solo tienen una copia parcial de la cadena de bloques y dependen de los nodos completos para verificar transacciones.

En resumen, un nodo es un componente importante en una red Blockchain y puede referirse a un dispositivo físico - PC, un programa de software o un punto de conexión en la red.

Red Blockchain con N nodos



Copyright https://www.freepik.es/vector-gratis/fondo-conexion-red-trabajo_1188386.htm

6.4. Diferencia de un nodo Ethereum y un nodo Sawtooth

Un nodo Ethereum y un nodo Sawtooth son dos tipos de nodos de Blockchain diferentes que utilizan diferentes algoritmos de consenso y tienen diferentes características y funcionalidades.

Ethereum es una plataforma de Blockchain descentralizada que utiliza un algoritmo de consenso de prueba de trabajo (PoW) para validar las transacciones y crear nuevos bloques. Los nodos Ethereum procesan transacciones y ejecutan contratos inteligentes en la red Ethereum. Los nodos Ethereum también pueden actuar como mineros, compitiendo para agregar nuevos bloques a la cadena de bloques y ganar recompensas en forma de ETH que es la criptomoneda conocido como GAS.

Por otro lado, Sawtooth es otra plataforma de Blockchain descentralizada que utiliza varios algoritmos de consenso para validar las transacciones y crear nuevos bloques. Los nodos Sawtooth procesan transacciones y mantienen la integridad de la cadena de bloques. Sawtooth es un proyecto de Hyperledger, una organización de código abierto que desarrolla tecnologías de Blockchain empresariales.

En resumen, la principal diferencia entre un nodo Ethereum y un nodo Sawtooth radica en su algoritmo de consenso, lo que afecta a cómo se validan las transacciones y se crean nuevos bloques en la cadena de bloques. Además, Ethereum es una plataforma de Blockchain pública, mientras que Sawtooth está diseñada para aplicaciones empresariales y tiene características específicas para su uso en entornos empresariales.

7. Smart Contract

Un Smart Contract o “contrato inteligente” es un programa informático que se ejecuta automáticamente en una Blockchain o cadena de bloques. Esencialmente, un Smart Contract es un conjunto de funcionalidades y condiciones que se establecen y que se ejecutan una vez que se cumplen las condiciones programadas en el contrato.

Estos contratos se ejecutan en una plataforma descentralizada y pueden utilizarse para automatizar y hacer cumplir acuerdos entre dos o más partes, sin necesidad de intermediarios como abogados, bancos u otros terceros.

Los Smart Contract se utilizan en muchas aplicaciones, desde pagos automatizados hasta sistemas de votación y de identificación digital. Debido a que los Smart Contract son ejecutados automáticamente en la Blockchain, son muy seguros y no pueden ser alterados una vez que se han establecido las condiciones.

“En la practica un Smart Contract son funcionalidades que se programan, responde a una entrada de datos para ejecutar una acción o devuelve una respuesta – datos”

Un Smart Contract es un conjunto de funcionalidades que se programan y se ejecutan automáticamente en la Blockchain. Estas funcionalidades definen las condiciones que deben cumplirse para que el contrato se ejecute y establecen la lógica de la transacción, determinando la entrada y la salida de datos.

Los Smart Contract son muy útiles porque eliminan la necesidad de intermediarios y aumentan la transparencia y la seguridad de las transacciones. Además, una vez que se establecen las condiciones del contrato, éstas no pueden ser alteradas, lo que garantiza la integridad del proceso.

En resumen, los Smart Contract son programas informáticos que establecen funcionalidades y condiciones, y se ejecutan automáticamente en la Blockchain cuando se cumplen dichas condiciones, lo que permite la automatización de procesos y la eliminación de intermediarios.

7.1. Desarrollar un Smart Contract

Para desarrollar un Smart Contract, es necesario tener conocimientos en programación y un entendimiento básico de cómo funciona una Blockchain. Para programar un Smart Contract debemos realizar estas acciones:

1. Elegir la plataforma Blockchain: Antes de empezar a programar un Smart Contract, es importante que elijas la plataforma Blockchain en la que desees implementarlo. Por ejemplo, Ethereum es una de las plataformas Blockchain más utilizadas para la creación de Smart Contracts, pero existen otras plataformas como Hyperledger Sawtooth, Corda o EOS que también son utilizadas.
2. Selecciona un lenguaje de programación: La mayoría de las plataformas Blockchain tienen sus propios lenguajes de programación para la creación de Smart Contracts. Por ejemplo, Ethereum utiliza Solidity, mientras que Hyperledger Sawtooth utiliza Go, Java o JavaScript. Asegúrate de elegir el lenguaje de programación adecuado para la plataforma Blockchain que has seleccionado.
3. Define las reglas del Smart Contract: Una vez que hayas seleccionado la plataforma Blockchain y el lenguaje de programación, es hora de definir las reglas del Smart Contract. Esto implica determinar las condiciones que deben cumplirse para que el contrato se ejecute y establecer la lógica de la transacción, determinando la entrada y la salida de datos.
4. Escribe el código: Con las reglas del Smart Contract definidas, es hora de escribir el código. Utiliza el lenguaje de programación seleccionado para escribir el código del Smart Contract, asegurándote de seguir las mejores prácticas de programación y cumplir con los estándares de la plataforma Blockchain.
5. Prueba el Smart Contract: Una vez que hayas escrito el código del Smart Contract, es importante que lo pruebes en un entorno de prueba antes de implementarlo en la Blockchain. Esto te permitirá detectar y corregir errores antes de que el contrato se implemente en la red.
6. Implementa el Smart Contract: Una vez que hayas probado el Smart Contract y estés seguro de que funciona correctamente, es hora de implementarlo en la plataforma Blockchain.
7. Monitorea y actualiza el Smart Contract: Una vez que el Smart Contract esté en funcionamiento, es importante que lo monitorees regularmente para asegurarte de que sigue funcionando correctamente. Además, es posible que necesites actualizar el contrato a medida que cambian las necesidades de tu aplicación.

En resumen, para desarrollar un Smart Contract, necesitas seleccionar la plataforma Blockchain adecuada, elegir el lenguaje de programación adecuado, definir las reglas del Smart Contract, programar el código, probarlo en un entorno de prueba, implementarlo en la Blockchain y monitorear y actualizar el contrato según sea necesario.

7.2. Programar un Smart Contract

Para crear un programar Smart Contract utilizando Hyperledger Sawtooth, seguimos las siguientes acciones:

1. Configurar el entorno de desarrollo: esto implica instalar los paquetes necesarios, configurar el entorno de ejecución y configurar el entorno de prueba. Para obtener más información sobre cómo configurar el entorno de desarrollo de Sawtooth, consulta la documentación oficial.
2. Programar las funcionalidades del Smart Contract: esto implica definir la lógica de programación de las funcionalidades que deben cumplir el contrato así como validar la entrada y la salida de datos.
3. Selecciona el lenguaje de programación: Sawtooth admite varios lenguajes de programación, incluyendo Python, JavaScript y Rust. Elige el lenguaje de programación que mejor se adapte a tus necesidades y habilidades.
4. Crea el Smart Contract: Utiliza el lenguaje de programación seleccionado para escribir el código del Smart Contract, asegurándote de seguir las mejores prácticas de programación y cumplir con los estándares de Sawtooth. Puedes utilizar el SDK de Sawtooth para crear el Smart Contract.
5. Prueba el Smart Contract: Una vez que hayas escrito el código del Smart Contract, es importante que lo pruebes en un entorno de desarrollo. Esto te permitirá detectar y corregir errores antes de que el contrato se implemente en la red.

6. Desplegar el Smart Contract: Una vez que hayas probado el Smart Contract y estés seguro de que funciona correctamente, es hora de instalarlo en producción.
7. Monitorea y actualiza el Smart Contract: Una vez que el Smart Contract esté en funcionamiento, es importante que lo monitorea regularmente para asegurarte de que sigue funcionando correctamente. Además, es posible que necesites actualizar el contrato a medida que cambian las necesidades de tu aplicación.

En resumen, para crear un Smart Contract utilizando Hyperledger Sawtooth, debes configurar el entorno de desarrollo, definir la lógica de programación de las funcionalidades del Smart Contract, seleccionar el lenguaje de programación, crear el Smart Contract, probarlo en un entorno de prueba, desplegar en producción, monitorear y actualizar el contrato según sea necesario.

8. Configurar entorno desarrollo

8.1 configurar el nodo Blockchain

Configurar un nodo Blockchain Hyperledger se puede realizar en una PC, portátil, en un servidor o en la nube (Amazon, Google, Azure, VPS etc.)

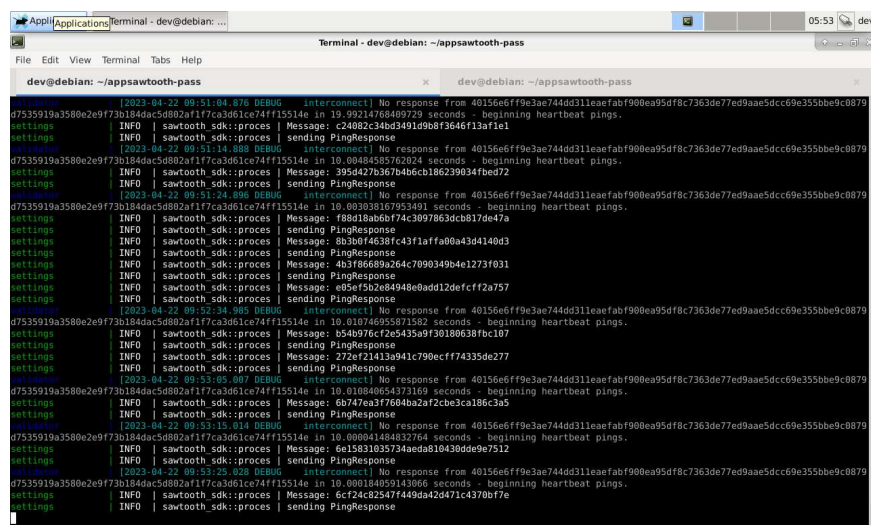
Para este desarrollo vamos a instalar Hyperledger Sawtooth en un ordenador portátil con sistema operativo DEBIAN Linux -v10 con estas características técnicas:

- Memoria RAM 8 Mbytes
- Disco duro de 64 Mbytes
- 4 cores

El software instalado es el siguiente:

- Git
- Docker
- Docker-compose
- NodeJS

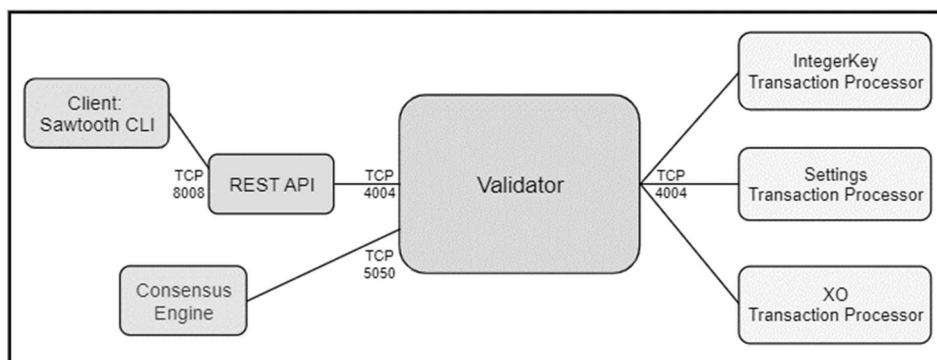
Los nodos Hyperledger Sawtooth los vamos a instanciar usando la herramienta [docker-compose](#), en la siguiente imagen visualizamos los nodos instanciados:



visualizamos los nodos instanciados y comprobamos el estado, todos están 'UP' – okay.

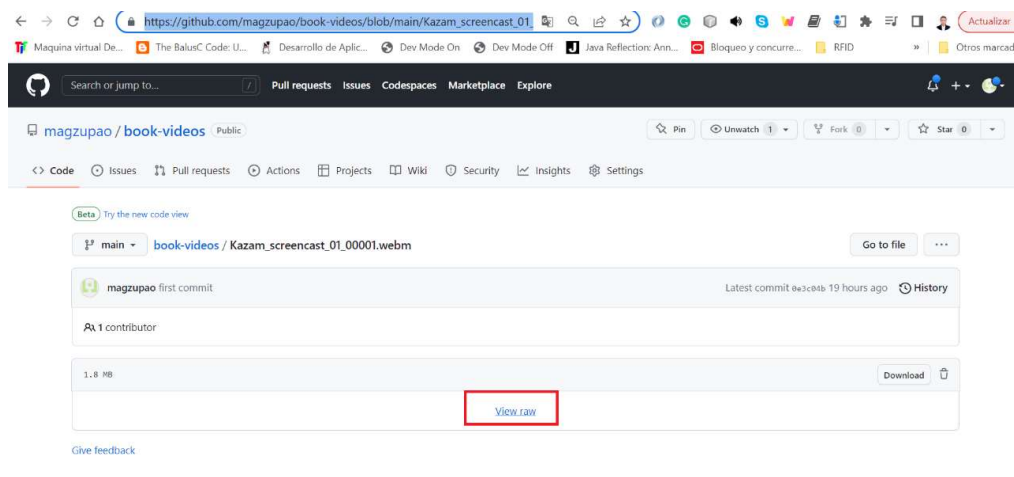
```
Applications Terminal - dev@debian: ... 05:54 dev
Terminal - dev@debian: ~/appsawtooth-pass
File Edit View Terminal Tabs Help
dev@debian: ~/appsawtooth-pass dev@debian: ~/appsawtooth-pass
dev@debian:~/appsawtooth-pass$ docker ps -a
CONTAINER ID   IMAGE                                     COMMAND                  CREATED        STATUS        PORTS
43bede7e3ac1   hyperledger/sawtooth-shell:chime        "bash -c 'sawtooth k..." 4 minutes ago  Up 4 minutes  4004/tcp, 8008/tcp
7698475c7b10   hyperledger/sawtooth-devmode-engine-rust:chime "devmode-engine-rust..." 4 minutes ago  Up 4 minutes
208748d5af2a   hyperledger/sawtooth-settings-tp:chime "settings-tp -vv -C ..." 4 minutes ago  Up 4 minutes  4004/tcp
8ced8e6ca29b   hyperledger/sawtooth-rest-api:chime      "sawtooth-rest-api -..." 4 minutes ago  Up 4 minutes  4004/tcp, 0.0.0.0:8008->8008/tcp, :::8008->8008/tcp
126bcb224fce   hyperledger/sawtooth-validator:chime     "bash -c 'sawadm key..." 4 minutes ago  Up 4 minutes  0.0.0.0:4004->4004/tcp, :::4004->4004/tcp
dev@debian:~/appsawtooth-pass$
```

El ordenador tiene desplegado los recursos que se define en la imagen de arquitectura:



https://github.com/magzupao/book-videos/blob/main/Kazam_screencast_01_00001.webm para descargar el video hay que dar 'View raw'.

Reproductor online <https://webm.to/player/?lang=es> si no se puede visualizar en el ordenador.



En el video, hasta el minuto se visualiza todo el despliegue del nodo Blockchain, después del minuto visualizamos el estado de todos los servicios que contiene el nodo, todo okay.

En esta dirección web puede tener varios ejemplos de como implementar un Smart Contract que en el lenguaje de Sawtooth es conocido como '[Transaction](#)'

9 Desarrollar Smart Contract

La funcionalidad a implementar en un Smart Contract será: **“venta de entradas”** para algún evento o servicio.

Enumeramos las funciones básicas a desarrollar:

- El usuario compra una entrada para el partido PERU vs ALEMANIA
- El usuario registra su compra ingresando su DNI y el EVENTO DEPORTIVO elegido
- Se procesa los datos y se registra la entrada la cadena Blockchain.
- Solo el usuario puede consultar su entrada usando su DNI

Entorno desarrollo

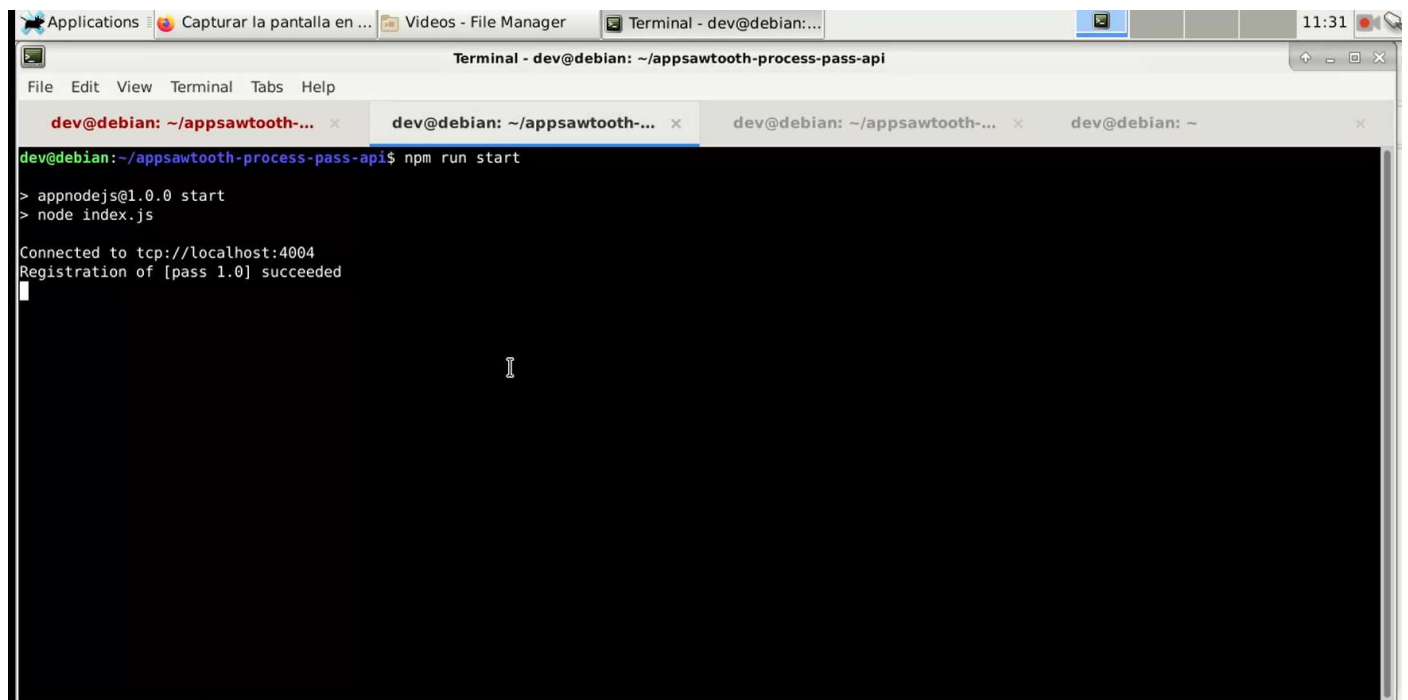
Nuestro entorno de desarrollo tiene las siguientes características técnicas:

- Sistema operativo Debian Linux v10, configuración básica (8 Mbytes de RAM)
- IDE VScode
- NodeJs v16.19.1
- Npm v8.19.3

https://github.com/magzupao/book-videos/blob/main/Kazam_screencast_01_00002.webm

Hasta el segundo 10 visualizamos la carga del Smart Contract, el contrato está identificado con el nombre:

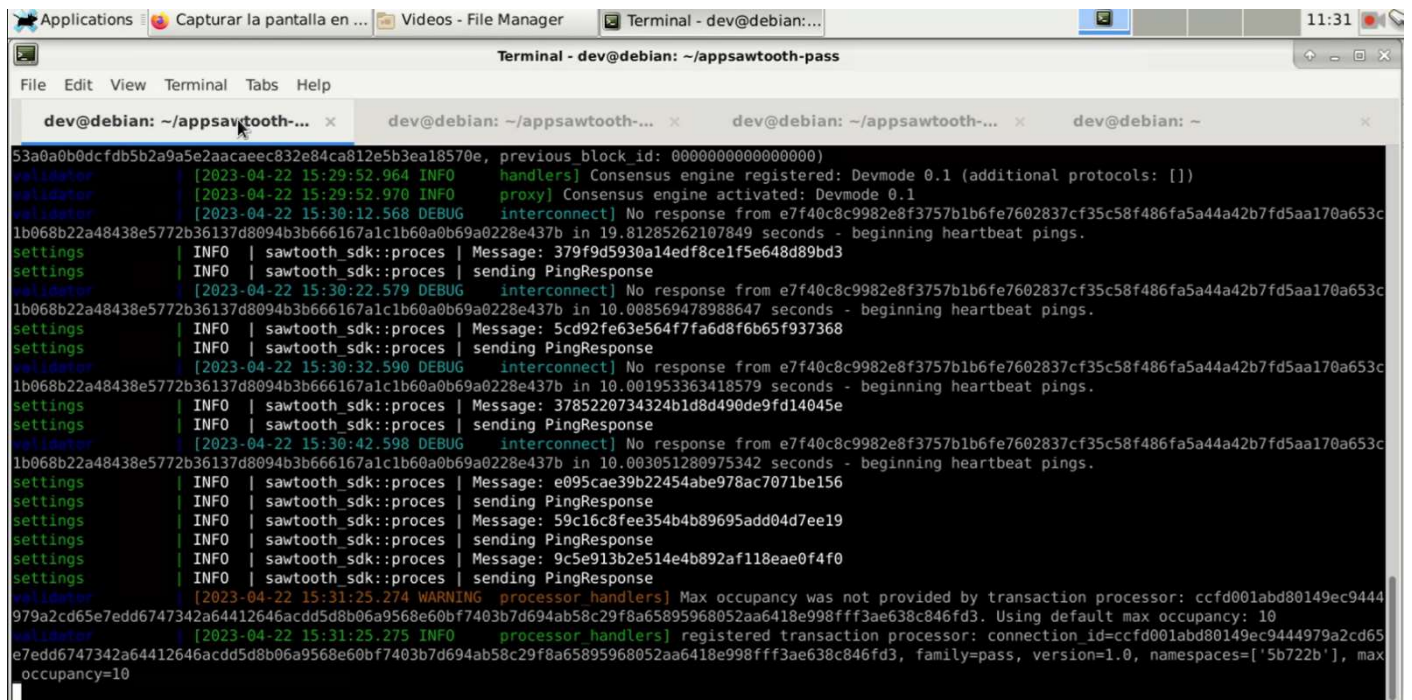
pass: 1.0



```
dev@debian: ~/appsawtooth-process-pass-api$ npm run start
> appnodejs@1.0.0 start
> node index.js
Connected to tcp://localhost:4004
Registration of [pass 1.0] succeeded
```

Después, visualizamos el registro del contrato en el nodo Blockchain, como vemos en la parte inferior de la imagen:

Family: pass, versión=1.0



```
dev@debian: ~/appsawtooth-pass
53a0a0b0dcfdb5b2a9a5e2aacaeec832e84ca812e5b3ea18570e, previous block id: 0000000000000000
[2023-04-22 15:29:52.964 INFO handlers] Consensus engine registered: Devmode 0.1 (additional protocols: [])
[2023-04-22 15:29:52.970 INFO proxy] Consensus engine activated: Devmode 0.1
[2023-04-22 15:30:12.568 DEBUG interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 19.81285262107849 seconds - beginning heartbeat pings.
[2023-04-22 15:30:22.579 DEBUG interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.008569478988647 seconds - beginning heartbeat pings.
[2023-04-22 15:30:32.590 DEBUG interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.001953363418579 seconds - beginning heartbeat pings.
[2023-04-22 15:30:42.598 DEBUG interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.003051280975342 seconds - beginning heartbeat pings.
[2023-04-22 15:31:25.274 WARNING processor handlers] Max occupancy was not provided by transaction processor: ccfd001abd80149ec9444
979a2cd65e7edd6747342a64412646acdd5d8b06a9568e60bf7403b7d694ab58c29f8a65895968052aa6418e998fff3ae638c846fd3. Using default max occupancy: 10
[2023-04-22 15:31:25.275 INFO processor handlers] registered transaction processor: connection_id=ccfd001abd80149ec9444979a2cd65
e7edd6747342a64412646acdd5d8b06a9568e60bf7403b7d694ab58c29f8a65895968052aa6418e998fff3ae638c846fd3, family=pass, version=1.0, namespaces=['5722b'], max
occupancy=10
```

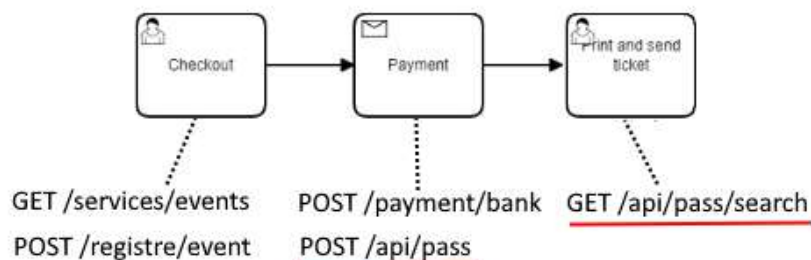
10.Utilizar Smart Contract

Para interactuar con el Smart Contract necesitamos de una interfaz, en este caso será una API que pueda interactuar con la aplicación.

El siguiente diagrama de flujo representa una aplicación o servicio web en producción que está activo e integrara con el nodo Blockchain a través de la API 'pass' que expone dos servicios:

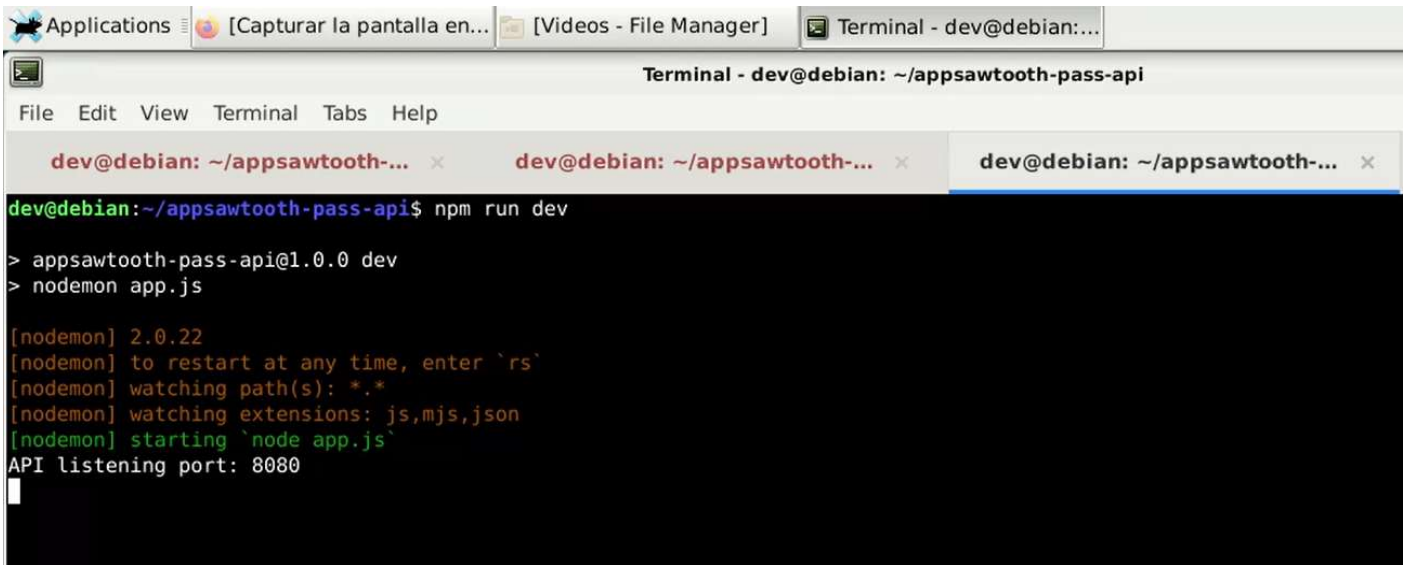
POST registramos el ticket en Blockchain

GET consultamos el ticket en Blockchain



Aplicación WEB en producción que se integra con Blockchain por medio de una API

Inicializamos la API pass:



```
dev@debian: ~/appsawtooth-pass-api
File Edit View Terminal Tabs Help

dev@debian: ~/appsawtooth-pass-api$ npm run dev

> appsawtooth-pass-api@1.0.0 dev
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
API listening port: 8080
```

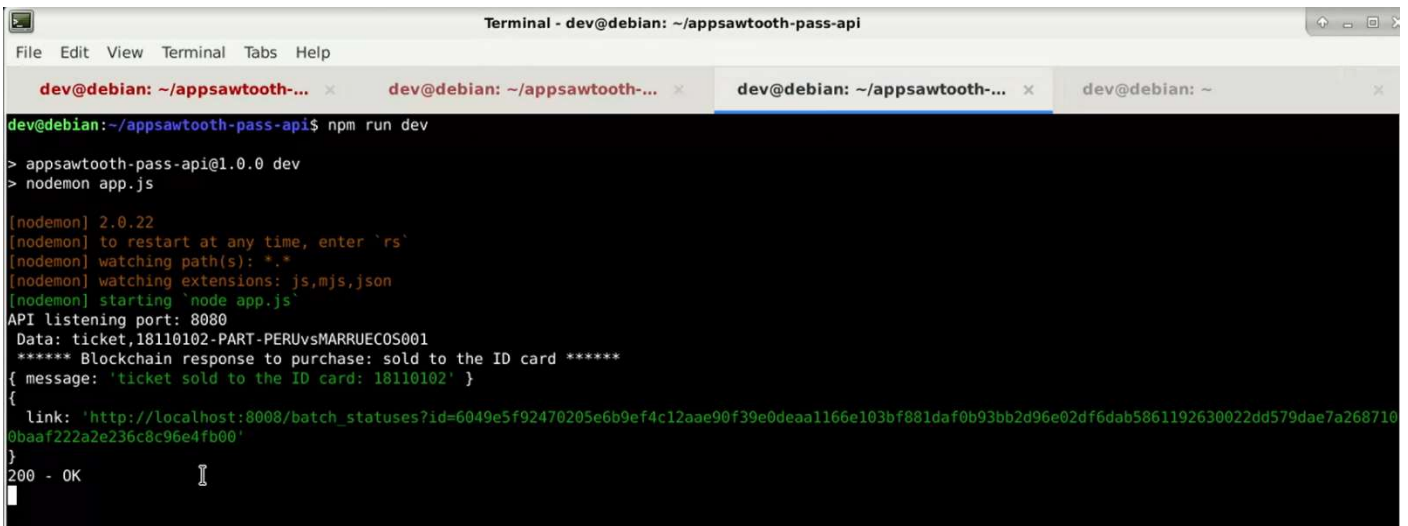
Registramos una entrada – POST, en la realidad esta llamada se invocará desde la aplicación web.



```
dev@debian: ~
File Edit View Terminal Tabs Help

dev@debian: ~$ curl -X POST http://localhost:8080/api/pass -H 'Content-Type: application/json' -d '{"dni":"18110102","event":"PART-PERUVsMARRUECOS001"}'
```

Obtenemos la respuesta de Blockchain:



```
dev@debian: ~/appsawtooth-pass-api
File Edit View Terminal Tabs Help

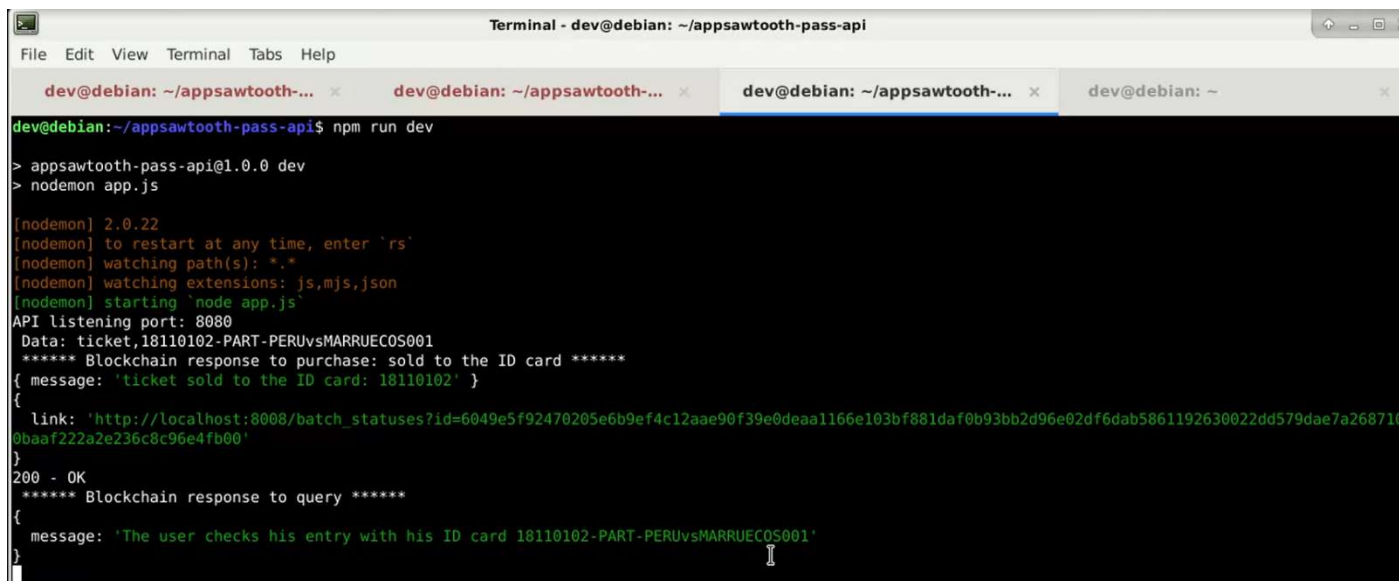
dev@debian: ~/appsawtooth-pass-api$ npm run dev

> appsawtooth-pass-api@1.0.0 dev
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
API listening port: 8080
Data: ticket,18110102-PART-PERUVsMARRUECOS001
***** Blockchain response to purchase: sold to the ID card *****
{ message: 'ticket sold to the ID card: 18110102' }
{
  link: 'http://localhost:8080/batch_statuses?id=6049e5f92470205e6b9ef4c12aae90f39e0dea1166e103bf881daf0b93bb2d96e02df6dab5861192630022dd579dae7a2687100baaf222a2e236c8c96e4fb00'
}
200 - OK
```

Consultamos el ticket creado con GET:

```
curl -X GET "http://localhost:8080/api/pass/search?dni=18110102"
```



```
Terminal - dev@debian: ~/appsawtooth-pass-api
File Edit View Terminal Tabs Help
dev@debian: ~/appsawtooth-... x dev@debian: ~/appsawtooth-... x dev@debian: ~/appsawtooth-... x dev@debian: ~ x
dev@debian:~/appsawtooth-pass-api$ npm run dev
> appsawtooth-pass-api@1.0.0 dev
> nodemon app.js

[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node app.js`
API listening port: 8080
Data: ticket,18110102-PART-PERUVsMARRUECOS001
***** Blockchain response to purchase: sold to the ID card *****
{ message: 'ticket sold to the ID card: 18110102' }
{
  link: 'http://localhost:8080/batch_statuses?id=6049e5f92470205e6b9ef4c12aae90f39e0deaa1166e103bf881daf0b93bb2d96e02df6dab5861192630022dd579dae7a2687100baaf222a2e236c8c96e4fb00'
}
200 - OK
***** Blockchain response to query *****
{
  message: 'The user checks his entry with his ID card 18110102-PART-PERUVsMARRUECOS001'
}
```

Los próximos capítulos contendrán:

- Desarrollo de NFT's
- Diseño y despliegue de una arquitectura Empresarial

RECURSOS

- <https://chat.openai.com>
- <https://es.lovepik.com/image-401719714/blockchain.html>
- Web Hyperledger: <https://www.hyperledger.org/use/sawtooth>
- <http://basenatos.blogspot.com/2009/02/ejercicio-4-sistema-de-vuelos.html>
- <https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-II-conceptos-basicos-proteccion-de-datos>
- <https://social.msdn.microsoft.com/Forums/sqlserver/es-ES/e7305f9d-9e2a-4c96-83a8-fd8d5de8920e/diagrama-entidad-relacion?forum=vcse>
- https://www.freepik.es/vector-gratis/fondo-conexion-red-trabajo_1188386.htm
- <https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-II-conceptos-basicos-proteccion-de-datos>
- <https://www.ledger.com/es/academy/que-son-las-claves-publicas-y-privadas>
- <https://101blockchains.com/es/hyperledger-or-ethereum/>
- <http://techdatasmex.blogspot.com/2018/03/las-5-mejores-plataformas-de-blockchain.html>
- <https://codigoonclick.com/mejores-lenguajes-programacion-para-2018/>
- https://sawtooth.hyperledger.org/docs/1.2/app_developers_guide/installing_sawtooth.html