UTILITIES

BLOCKCHAIN

Using of the tool "Hyperledger Sawtooth"

2023

Ing. Marco Aurelio Guado Zavaleta
Master in Corporate Software Development
https://www.linkedin.com/in/marcoguado/

INTRODUCTION

Blockchain is a distributed ledger technology that allows multiple parties to have access to a shared database, without the need for a central authority to control it. The technology was popularized by the Bitcoin cryptocurrency, but today it is used in a wide range of applications, from voting systems to supply chains.

Hyperledger Sawtooth is an open source Blockchain platform developed by the Linux Foundation. It was designed to be modular and scalable, making it suitable for complex enterprise applications. Sawtooth uses a variety of consensus algorithms to validate transactions and create new blocks on the blockchain. It also includes advanced features, such as the ability to implement smart contracts in multiple programming languages and granular permissions management for network users. In summary, Hyperledger Sawtooth is a powerful and flexible blockchain platform that can be tailored to a company's specific needs.

This document describes what is needed to implement a 'Smart Contract' as:

- Installing a Blockchain Node.
- Developing a Smart Contract.
- Using the 'Smart Contract' through an API (Application Programming Interface).

INDEX

1. Definition

Blockchain is a distributed recording technology used to store and verify transactions in a decentralized network. It is based on a decentralized and secure database, where data is recorded in interconnected and encrypted blocks to ensure its integrity and security.

2. Descentralized database

A decentralized database stores and distributes information (data) across multiple nodes or devices, rather than being centralized on a single server or location. centralized on a single server or location. In this type of database, each node has a full copy of the data and updates are propagated of the data and updates are propagated across the network to keep all copies synchronized.

The advantage of a decentralized database is that it offers greater resilience to failure and increased security, since there is no single point of failure or attack. In addition, a decentralized database can allow for collaboration and multi-user collaboration and participation of multiple users without the need for a central authority, which can be useful in applications where transparency and trust are required, as there is no single point of failure or attack. applications where transparency and trust are required, such as blockchain applications.

2.1. Characteristics

The characteristics of a decentralized database are:

- Distribution: Information is stored and distributed across multiple nodes or devices, which means that there is there is no single point of failure or attack.
- Redundancy: Each node has a complete copy of the data, which means that, in the event of a node node fails, the information will still be available on other nodes.
- Security: Security and privacy measures can be implemented on each node individually to protect information. protect the information.
- Transparency: Information can be transparently accessed and updated by all users who have access to the database. who have access to the database.
- Participation: Users can participate in the database without the need for a central authority, which can be useful in applications where the database is which can be useful in applications where trust and transparency are required.
- Synchronization: Updates are propagated across the network to keep all copies of the data in sync. synchronized.
- Scalability: The decentralized database can scale efficiently as more nodes are added to the network. nodes are added to the network.

In summary, a decentralized database provides greater resiliency to failure and increased security, and enables multi-user collaboration and participation without the need for a central authority.

3. Database (centralized)

A centralized database stores information on a single server or centralized location. In this type of database, users access the data through a network, and all updates and modifications are made on the central server.

The advantage of a centralized database is that it can be easier to manage and maintain, since all data is in one place and security and privacy measures can be implemented at a single point. However, it also has some disadvantages, such as lack of redundancy and the possibility of a single point of failure, which can increase the risk of data loss. failure, which can increase the risk of data loss and system failures.

In addition, in a centralized database, a central authority may be required to manage and control access to the data, which can increase the risk of data loss and system failure. control access to the data, which can limit transparency and participation by multiple users.

3.1. Characteristics

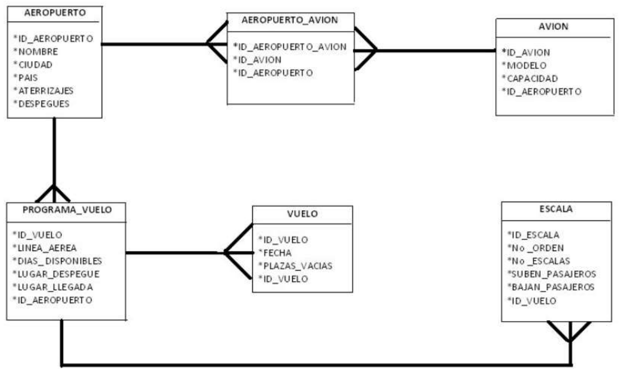
The characteristics of a centralized database are:

- Centralization: Information is stored on a single server or centralized location.
- Administration: The administration and maintenance of the database is done in one place, which makes it easier to manage facilitates its management.
- Access control: Access control to the database is performed in the central server, which allows the implementation of security and privacy security and privacy measures can be implemented effectively.
- Limited scalability: The centralized database has limited ability to scale as more data and users are added. and users are added.
- Vulnerability to failure: A single point of failure can result in the loss of all information stored in the database.
- Dependence on network connection: The network connection is essential for accessing the database, which can limit the availability of information if it is not available.
- Dependence on central authority: The centralized database depends on a central authority to manage and control access to the data. and control access to the data, which can limit transparency and user participation.

In summary, a centralized database offers centralized administration and access control, but has limitations in terms of scalability, vulnerability to failure and dependence on network connection and central authority.

Examples of Centralized and Decentralized Databases:

Databases (centralized) store information in structures known as rows and columns (similar to Excel files). These structures are called tables and there can be N tables in a database that represent the data model of a business. For example, in the following image we represent the data model of an airport (seen in a simple way).

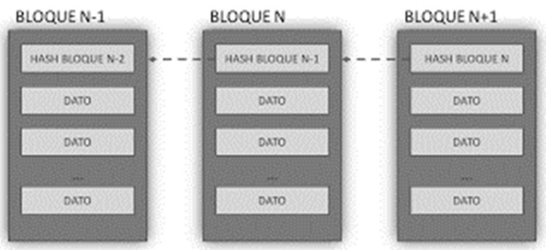


Copyright http://basenatos.blogspot.com/2009/02/ejercicio-4-sistema-de-vuelos.html

Under this model we can consult in the database; which types of flights have stopovers, which flights use which type of aircraft, which airports have scheduled flights at night, etc.

In summary, the (centralized) Databases store the data under a schema that groups N tables in this case; the Tables are; Airport, Flight, Scala etc.

Blockchain being a distributed database stores information sequentially as if it were a train, but only stores extremely useful and shareable information.



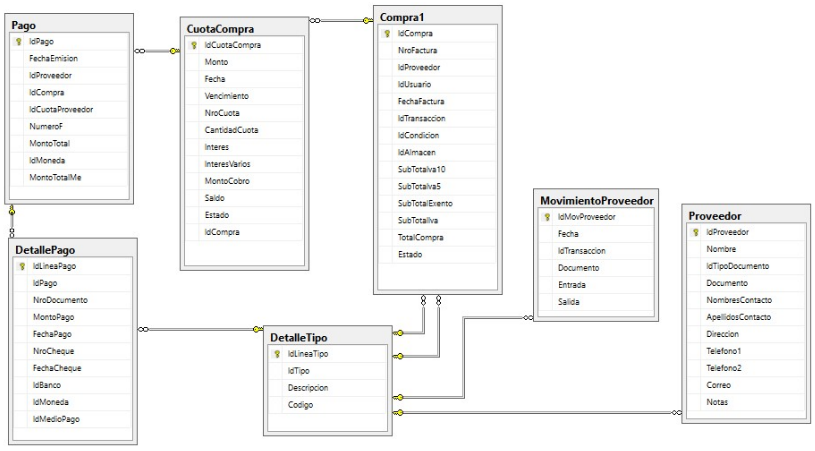
Copyright https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-II-conceptos-basicos-proteccion-de-datos

What data could we store following the Airport example, it could be:

DATA = "Flight KL 744, AMS - LIM, pasj 234, fuel 20 tn, Jorge Chavez Airport."

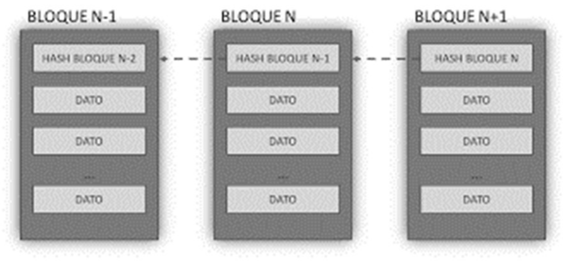
Blockchain is a distributed database that stores data in a different way, it is not in structure but in sequential order.

Let's see another example of a Database (centralized) payment system, with several tables:



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

But in a Blockchain storage system for PAYMENTS, it would only be useful to record the payment.



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

DATA = "PAYMENT, TotalAmount=123,NoFee=2,Date=12/2/2023"

Again, Blockchain chains only store information that is important for the business, in this case the payment is stored in date and amount and that the information will not be manipulated and also we have the security that we can share it, for example, for audit issues.

Another example of the use of Blockchain services is in the sale of tickets since the ticket cannot be copied or altered. altered, the data to be stored would be something like this:

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

If the input is consumed, the information would change and would be recorded in this way:

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

With this we manage to disable the entrance.

"All these examples discussed, are under the Smart Contract type Blockchain services as they need to execute certain rules before storing the information."

Note. - in practice a "Smart Contract" are programming lines that implement a functionality, for example, in order to have the DATO record we can program by copying the information from the tables; event, user, payment and with that data we create a single record (DATO) that we are going to store in the Blockchain.

4. Recording data in Blockchain

To register the data in Blockchain the user must do it through a private certificate and can delegate his public certificate to other users so that they can see the information, this is one of the most important utilities that ensure that the information will never be manipulated.

In the (centralized) databases, users have permissions to write and read, while in Blockchain, users write using the private certificate and read using the public certificate. private certificate and read using the public certificate..

All users interacting with the Blockchain must do so using both their private and public certificates. Without certificates, the information cannot be accessed.

4.1. Private and public certificates

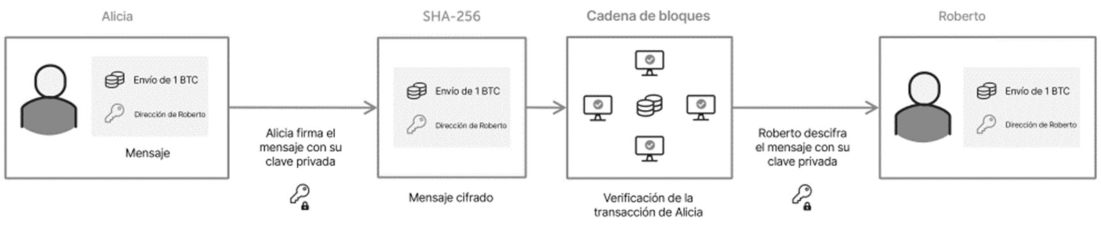
Private and public certificates are key components of the Blockchain infrastructure and are used to authenticate the identity of a user or system in order to interact in the Blockchain. Certificates are usually physical files that have stored in encrypted form the user's data, these certificates are issued by those who manage the Blockchain infrastructure in which they interact.
The following image graphs the usefulness of certificates, although the graph represents cryptocurrencies, the procedure is the same for a data. procedure is the same for a data.

1. Sent a datum, for example, that of the incoming purchase:

   Message => DATA =" PART-PERUvsALEMANIA-234N565M4-02/03/2023"

   And to validate that it has been the user Roberto, he signs it using his private certificate (private key).

2. The signed message is encrypted and passed to the Blockchain.
3. The user Roberto and only him using his public certificate (public key) can read the content of the information. information.



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

Note - no hacker has breached the security of Blockchain, in all the news to date that has been published about the theft of Blockchain, hackers have seized certificates (this being its Achilles heel). about Blockchain theft, hackers have taken over certificates (this being their Achilles heel). Blockchain.
But to correct this security point several strategies are being followed, one is to delegate to each user the custody of their certificates or two is to have a single user validating all transactions.

5. Blockchain services

Blockchain technology can be useful to implement 3 services:

- Cryptocurrencies.
- Smart contracts known as: "Smart Contract".
- NFS tokens that serve to identify the authenticity of a digital file.

Question: Is Bitcoin Blockchain? Bitcoin is a service that uses Blockchain technology and is in the category of cryptocurrencies.

Apart from using Blockchain as a cryptocurrency generator, there are other services such as "Smart Contracts", which is what is used in most Blockchain systems when processing data. which is what is used in most Blockchain systems when processing data, finally, we have the NFTs that together with the "Smart Contract" are used to ensure the authenticity and originality of some digital content. If we design a logo and distribute it on the Internet afterwards, how can we claim copyright? This is achieved by joining NFT's + Smart Contract.

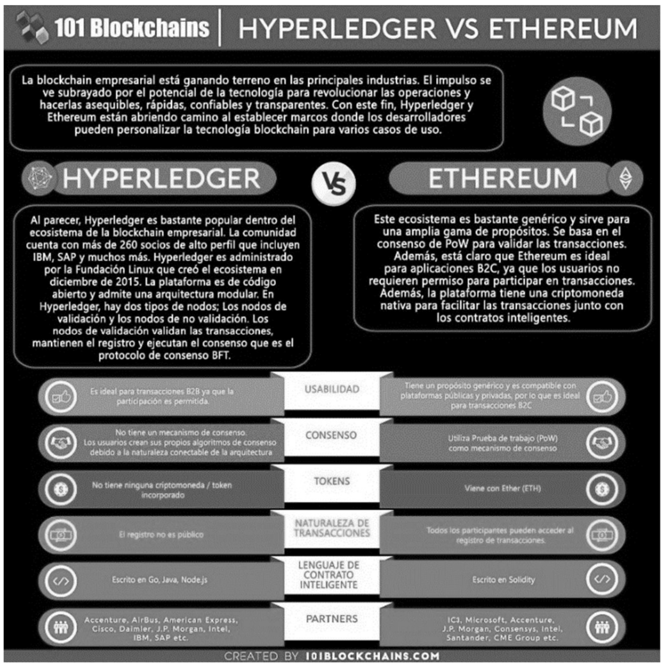
Another example is in audit issues where we can store documents ensuring their authenticity and ensuring that they have not been tampered with to date. that they have not been manipulated up to that date. In criminalistics using Blockchain technology is very useful to preserve the data of an evidence. the data of an evidence.

In summary: Blockchain technology is very useful when manipulating data and we want to ensure that those values will not be altered and we are confident that we can share information with third parties. be altered and we have the confidence to share the information to third parties.

Authenticity and trust are the great values of Blockchain technology

5.1. Types of Blockchain

Not all Blockchain platforms have a currency associated with them. In this case Hyperledger Sawtooth to encrypt data does not need a currency, on the other hand Ethereum having a currency is part of its business for each transaction (encrypt) uses a currency. In the following image we can see a comparative chart between Hyperledger and Ethereum.



101 Blockchains | HYPERLEDGER VS ETHEREUM

La blockchain empresarial está ganando terreno en las principales industrias. El impulso se ve subrayado por el potencial de la tecnología para revolucionar las operaciones y hacerlas asequibles, rápidas, confiables y transparentes. Con este fin, Hyperledger y Ethereum están abriendo camino al establecer marcos donde los desarrolladores pueden personalizar la tecnología blockchain para varios casos de uso.

HYPERLEDGER VS ETHEREUM

Al parecer, Hyperledger es bastante popular dentro del ecosistema de la blockchain empresarial. La comunidad cuenta con más de 260 socios de alto perfil que incluyen IBM, SAP y muchos más. Hyperledger es administrado por la Fundación Linux que creó el ecosistema en diciembre de 2015. La plataforma es de código abierto y admite una arquitectura modular. En Hyperledger, hay dos tipos de nodos; Los nodos de validación y los nodos de no validación. Los nodos de validación validan las transacciones, mantienen el registro y ejecutan el consenso que es el protocolo de consenso BFT.

Este ecosistema es bastante genérico y sirve para una amplia gama de propósitos. Se basa en el consenso de PoW para validar las transacciones. Además, está claro que Ethereum es ideal para aplicaciones B2C, ya que los usuarios no requieren permiso para participar en transacciones. Además, la plataforma tiene una criptomoneda nativa para facilitar las transacciones junto con los contratos inteligentes.

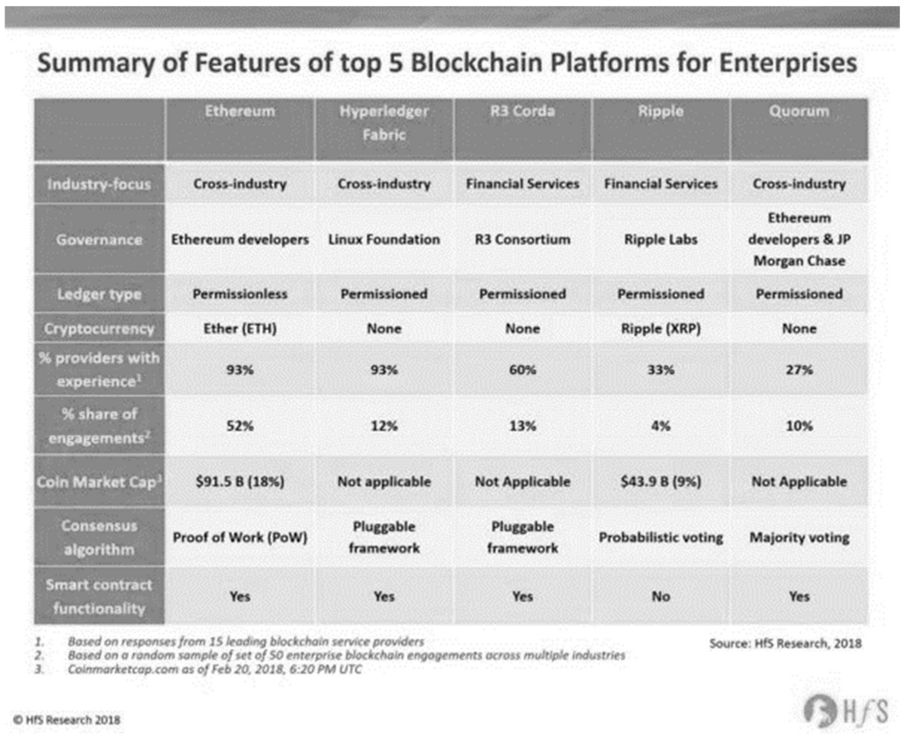
HYPERLEDGER		ETHEREUM
Es ideal para transacciones B2B ya que la participación es permitida.	USABILIDAD	Tiene un propósito genérico y es compatible con plataformas públicas y privadas, por lo que es ideal para transacciones B2C
No tiene un mecanismo de consenso. Los usuarios crean sus propios algoritmos de consenso debido a la naturaleza conectable de la arquitectura	CONSENSO	Utiliza Prueba de trabajo (PoW) como mecanismo de consenso
No tiene ninguna criptomoneda / token incorporado	TOKENS	Viene con Ether (ETH)
El registro no es público	NATURALEZA DE TRANSACCIONES	Todos los participantes pueden acceder al registro de transacciones.
Escrito en Go, Java, Node.js	LENGUAJE DE CONTRATO INTELIGENTE	Escrito en Solidity
Accenture, AirBus, American Express, Cisco, Daimler, J.P. Morgan, Intel, IBM, SAP etc.	PARTNERS	IC3, Microsoft, Accenture, J.P. Morgan, Consensys, Intel, Santander, CME Group etc.

CREATED BY 101BLOCKCHAINS.COM

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

6. Blockchain Platforms

To program using Blockchain technology there are many platforms, some of them are shown in the following graphic, but there are many more.



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://techdatatsmex.blogspot.com/2018/03/las-5-mejores-plataformas-de-blockchain.html

As you will see in the image above only two tools have Ethereum and Ripple cryptocurrencies.

Note - there is no need to use a coin to encrypt data in the Hyperledger Sawtooth Blockchain, if our business apart from encrypting data we want to generate a coin then we can use Ethereum.

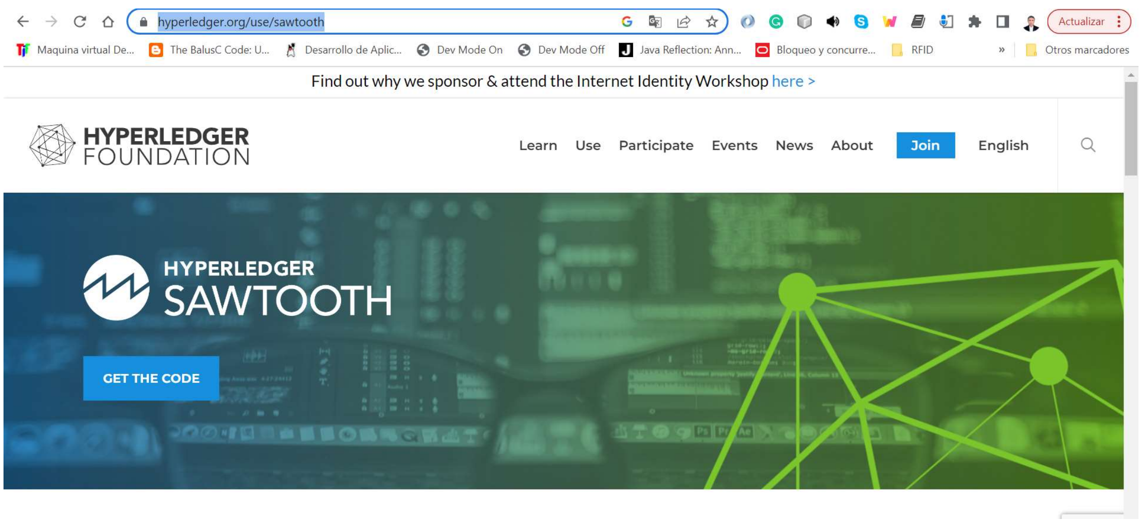
We can compare the above picture with the following comment:" if we want to program web pages, how many programming languages exist". Many that are represented in this image:

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

¡To program a web page, I have to learn all those programming languages, of course not! It is enough to choose a language.

The same goes for Blockchain, for this guide we are going to choose the Hyperledger Sawtooth platform.

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



6.1. Hyperledger Sawtooth

Hyperledger Sawtooth is an open source Blockchain technology platform developed by the Linux Foundation as part of the Hyperledger project. Linux as part of the Hyperledger project. Sawtooth is designed to provide a scalable and flexible enterprise platform for Blockchain applications. scalable and flexible enterprise platform for Blockchain applications.

The platform is based on a modular architecture and allows the implementation of decentralized applications with a wide range of performance and privacy requirements. Sawtooth uses several consensus algorithms, it is also scalable and consumes less power than some of the other consensus algorithms.

Sawtooth offers a number of features and tools that enable the creation and deployment of custom Blockchain applications, including a flexible smart contract language, a command-line interface for node administration, a web dashboard for network monitoring and management, and a RESTful API for integrating Blockchain applications with other enterprise applications.

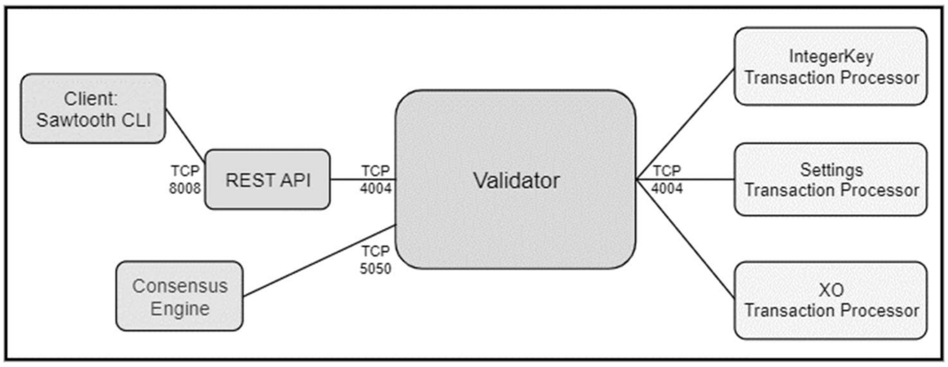
6.2. Hyperledger Sawtooth Feactures

Hyperledger Sawtooth is an open source Blockchain platform that is designed to be modular, scalable and secure. Here I present to you some of its most outstanding features:

- Modular architecture: Sawtooth uses a modular architecture, which means that the different layers of the Blockchain protocol, such as consensus, transaction validation and data storage, can be replaced and customized according to specific application needs.
- Dynamic consensus: Sawtooth allows the use of multiple consensus algorithms, allowing developers to choose the algorithm that best suits the needs of their application. In addition, Sawtooth uses a transaction-based consensus model, which means that transactions are grouped in blocks and undergo a validation process before being accepted on the Blockchain.
- Flexible data model: Sawtooth uses a flexible data model that allows the creation of different types of records, such as smart contracts and digital assets, which makes it easier for types of records, such as smart contracts and digital assets, making it easier for developers to create specific applications.
- Security: Sawtooth uses advanced security techniques such as transaction encryption and user authentication to protect the integrity of the Blockchain.
- Scalability: Sawtooth is designed to scale horizontally, which means that capacity can be added to the network by simply adding more nodes. capacity to the network by simply adding more nodes. In addition, Sawtooth uses a stateful partitioning model that allows for a transaction state partitioning model that enables transaction parallelization, allowing the network to handle a higher volume of transactions.

In summary, Hyperledger Sawtooth is a flexible and scalable Blockchain platform that allows developers to customize the Blockchain architecture and choose the consensus algorithm that best suits their application needs. In addition, Sawtooth uses advanced security techniques and is designed to scale horizontally to handle higher transaction volume.

Architecture of a Hyperledger Sawtooth Blockchain Node



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

6.3. ¿What is a nodo?

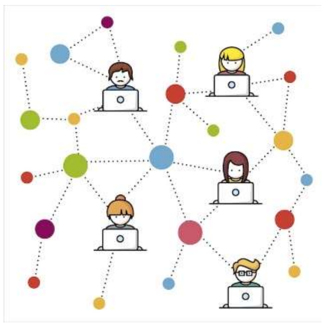
A Blockchain node is an essential component of a decentralized Blockchain network that helps maintain the integrity and security of the blockchain. A node can be defined as any device connected to the Blockchain network that participates in the process of validating transactions and confirming new blocks on the chain.

Each node in the network has a complete copy of the blockchain, meaning that it contains all data and transaction records since the creation of the chain. The nodes work together to validate transactions and confirm new blocks, meaning that if one of the nodes in the network detects an invalid or suspicious transaction, the other nodes can verify it and determine whether it should be accepted or rejected.

Blockchain nodes can be divided into different categories, depending on their function. For example, full nodes have a complete copy of the blockchain and participate in the validation of transactions and the creation of new blocks, while light nodes only have a copy of the blockchain. transactions and the creation of new blocks, while light nodes only have a partial copy of the blockchain and depend on the partial copy of the blockchain and rely on full nodes to verify transactions.

In summary, a node is an important component in a Blockchain network and can refer to a physical device - PC, a software program or a connection point in the network.

Blockchain network with N nodes



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

6.4. Difference between an Ethereum node and a Sawtooth node

An Ethereum node and a Sawtooth node are two different types of Blockchain nodes that use different consensus algorithms and have different features and functionalities.

Ethereum is a decentralized Blockchain platform that uses a proof-of-work (PoW) consensus algorithm to validate transactions and create new blocks. Ethereum nodes process transactions and execute smart contracts on the Ethereum network. Ethereum nodes can also act as miners, competing to add new blocks to the blockchain and earn rewards in the form of ETH which is the cryptocurrency known as GAS.

On the other hand, Sawtooth is another decentralized Blockchain platform that uses various consensus algorithms to validate transactions and create new blocks. Sawtooth nodes process transactions and maintain the integrity of the blockchain. integrity of the blockchain. Sawtooth is a project of Hyperledger, an open-source organization that develops enterprise blockchain technologies.

In short, the main difference between an Ethereum node and a Sawtooth node lies in their consensus algorithm, which affects how transactions are validated and new blocks are created on the blockchain. In addition, Ethereum is a public Blockchain platform, while Sawtooth is designed for enterprise applications and has specific features for its use enterprise applications and has specific features for use in enterprise environments.

7. Smart Contract

A Smart Contract or "smart contract" is a computer program that executes automatically on a Blockchain. Essentially, a Smart Contract is a set of functionalities and conditions that are established and executed once the conditions programmed into the contract are met.

These contracts are executed on a decentralized platform and can be used to automate and enforce agreements between two or more parties, without the need for intermediaries such as lawyers, banks or other third parties.

Smart contracts are used in many applications, from automated payments to voting systems and digital and digital identification systems. Because Smart Contracts are automatically executed on the Blockchain, they are highly secure and cannot be altered once the terms are set.

“In practice a Smart Contract is functionality that is programmed, responds to a data input to execute an action or returns a response - data.”

A Smart Contract is a set of functionalities that are automatically programmed and executed on the Blockchain. These functionalities define the conditions that must be met in order for the contract to be executed and establish the transaction logic, determining the input and output of data.

Smart Contracts are very useful because they eliminate the need for intermediaries and increase the transparency and security of transactions. In addition, once the terms of the contract are established, they cannot be altered, ensuring the integrity of the process.

In short, Smart Contracts are computer programs that establish functionalities and conditions, and are automatically executed on the Blockchain when these conditions are met, allowing the automation of processes and the elimination of intermediaries.

7.1. Developing a Smart Contract

To develop a Smart Contract, it is necessary to have programming knowledge and a basic understanding of how a Blockchain works. To program a Smart Contract we must perform the following actions:

1. Choose the Blockchain platform: Before you start programming a Smart Contract, it is important that you choose the Blockchain platform on which you want to implement it. choose the Blockchain platform on which you want to implement it. For example, Ethereum is one of the most used Blockchain platforms for the creation of Smart Contracts. most used Blockchain platforms for the creation of Smart Contracts, but there are other platforms such as Hyperledger such as Hyperledger Sawtooth, Corda or EOS that are also used.
2. Select a programming language: Most Blockchain platforms have their own programming languages for the creation of Smart Contracts. programming languages for the creation of Smart Contracts. For example, Ethereum uses Solidity, while Hyperledger Sawtooth uses Go, Java or JavaScript. Make sure you choose the right programming language programming language suitable for the Blockchain platform you have selected.

3. Define the rules of the Smart Contract: Once you have selected the Blockchain platform and programming language, it is time to define the rules of the Smart Contract. programming language, it's time to define the Smart Contract rules. This involves determining the conditions conditions that must be met for the contract to be executed and establish the transaction logic, determining the input and output of data. the data input and output.
4. Write the code: With the Smart Contract rules defined, it is time to write the code. Use the selected selected programming language to write the Smart Contract code, making sure to follow best programming practices and comply with programming best practices and comply with the Blockchain platform standards.
5. Test the Smart Contract: Once you have written the Smart Contract code, it is important to test it in a test environment. test it in a test environment before implementing it on the Blockchain. This will allow you to detect and correct errors before the contract is implemented on the network.
6. Implement the Smart Contract: Once you have tested the Smart Contract and are sure that it works correctly, it is time to implement it on the Blockchain. works correctly, it is time to implement it on the Blockchain platform.
7. Monitor and update the Smart Contract: Once the Smart Contract is up and running, it is important to monitor it regularly to ensure that it is working properly. important that you monitor it regularly to make sure it continues to function properly. In addition, you may need to update the contract as the needs of your application.

In summary, to develop a Smart Contract, you need to select the right Blockchain platform, choose the right appropriate programming language, define the Smart Contract rules, program the code, test it in a test environment, deploy it on the Blockchain, and monitor and update the contract as needed.

7.2. Programming a Smart Contract

To create a Smart Contract program using Hyperledger Sawtooth, follow these steps:

1. Configure the development environment: this involves installing the necessary packages, configuring the runtime environment and configuring the test environment. For more information on how to set up the Sawtooth development environment, please refer to the official documentation.
2. Program the smart contract functionalities: this involves defining the programming logic of the functionalities to be fulfilled by the contract, as well as validating the data input and output.
3. Select the programming language: Sawtooth supports several programming languages, such as Python, JavaScript and Rust. Choose the programming language that best suits your needs and skills.
4. Create the smart contract: Use the selected programming language to write the smart contract code, making sure to follow programming best practices and comply with Sawtooth standards. You can use the Sawtooth SDK to create the Smart Contract.
5. Test the Smart Contract: Once you have written the Smart Contract code, it is important that you test it in a development environment. This will allow you to detect and fix bugs before the contract is deployed on the network.
6. Deploy the Smart Contract: Once you have tested the Smart Contract and are confident that it works correctly, it is time to deploy it to production.
7. Monitor and update the Smart Contract: Once the Smart Contract is up and running, it is important that you monitor it regularly to ensure that it continues to function properly. In addition, you may need to update the contract as your application needs change.

In summary, to create a Smart Contract using Hyperledger Sawtooth, you must set up the development environment, define the programming logic of the Smart Contract functionalities, select the programming language, create the Smart Contract, test it in a test environment, deploy it to production, monitor and update the contract as needed.

8. Configure development environment

8.1 configuring the Blockchain node

Setting up a Blockchain Hyperledger node can be done on a PC, laptop, on a server or in the cloud (Amazon, Google, Azure, VPS etc.)

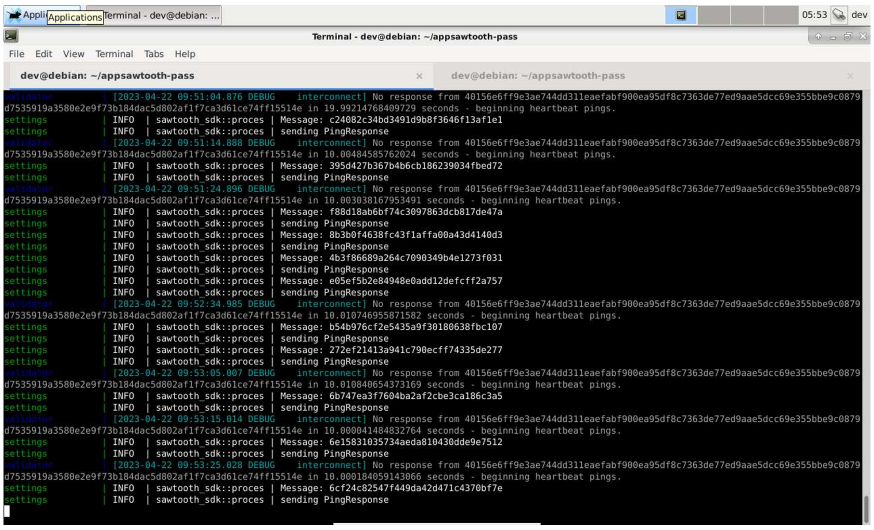
For this development we are going to install Hyperledger Sawtooth on a laptop with DEBIAN operating system Linux - v10 with these technical characteristics:

- RAM Memory 8 Mb Bbytes
- Disco duro de 64 Mb Bytes
- 4 cores

The software installed is as follows:

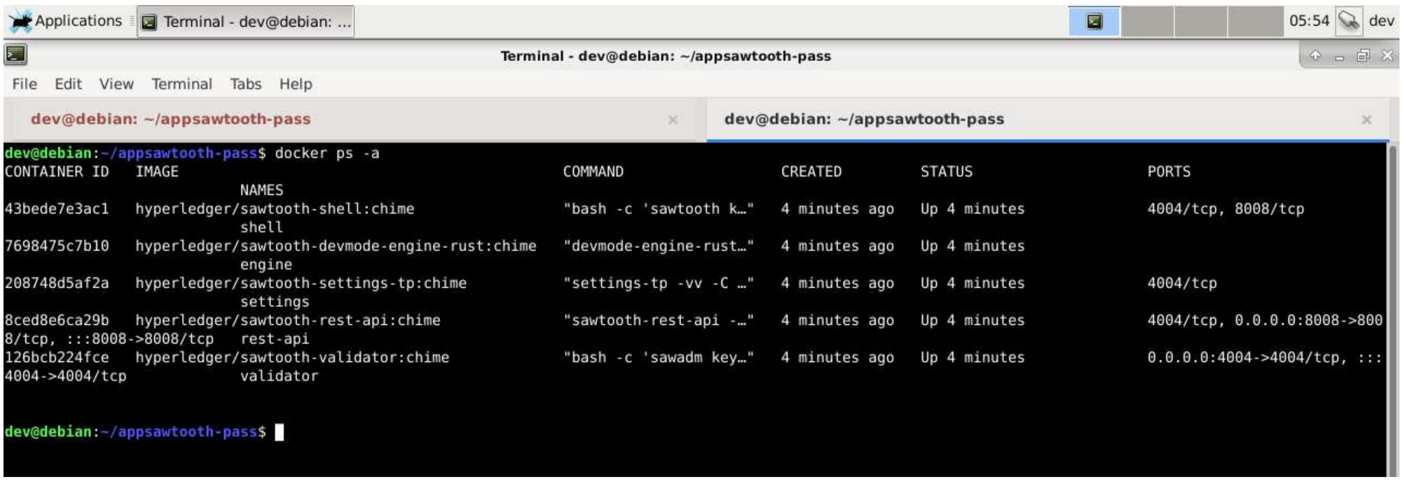
- Git
- Docker
- Docker-compose
- NodeJs

We are going to instantiate the Hyperledger Sawtooth nodes using the docker-compose tool, in the following image image we visualize the instantiated nodes:



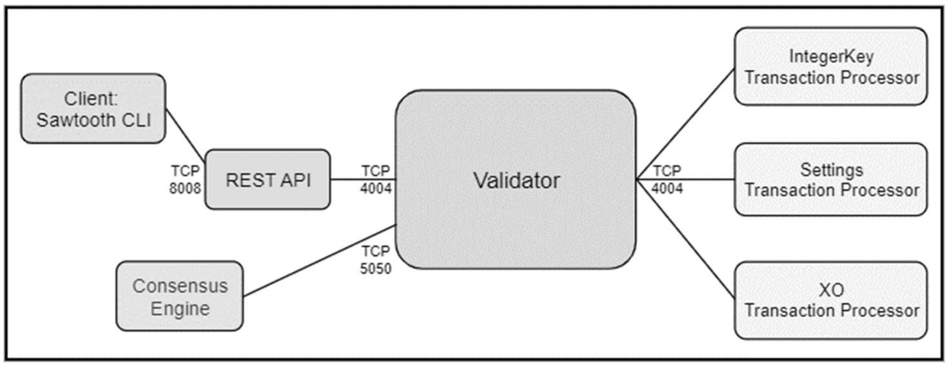
```
validator           | [2023-04-22 09:51:04.876 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 19.99214768409729 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: c24082c34bd3491d9b8f3646f13af1e1
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:51:14.888 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.00484585762024 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: 395d427b367b4b6cb186239034fbed72
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:51:24.896 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.003038167953491 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: f88d18ab6bf74c3097863dcb817de47a
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
settings            | INFO  | sawtooth_sdk::proces | Message: 8b3b0f4638fc43f1affa00a43d4140d3
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
settings            | INFO  | sawtooth_sdk::proces | Message: 4b3f86689a264c7090349b4e1273f031
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
settings            | INFO  | sawtooth_sdk::proces | Message: e05ef5b2e84948e0add12defcff2a757
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:52:34.985 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.010746955871582 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: b54b976cf2e5435a9f30180638fbc107
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
settings            | INFO  | sawtooth_sdk::proces | Message: 272ef21413a941c790ecff74335de277
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:53:05.007 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.010840654373169 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: 6b747ea3f7604ba2af2cbe3ca186c3a5
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:53:15.014 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.000041484832764 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: 6e15831035734aeda810430dde9e7512
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
validator           | [2023-04-22 09:53:25.028 DEBUG    interconnect] No response from 40156e6ff9e3ae744dd311eaefabf900ea95df8c7363de77ed9aae5dcc69e355bbe9c0879
d7535919a3580e2e9f73b184dac5d802af1f7ca3d61ce74ff15514e in 10.000184059143066 seconds - beginning heartbeat pings.
settings            | INFO  | sawtooth_sdk::proces | Message: 6cf24c82547f449da42d471c4370bf7e
settings            | INFO  | sawtooth_sdk::proces | sending PingResponse
```

We visualize the instantiated nodes and check the status, all are 'UP' - ok.



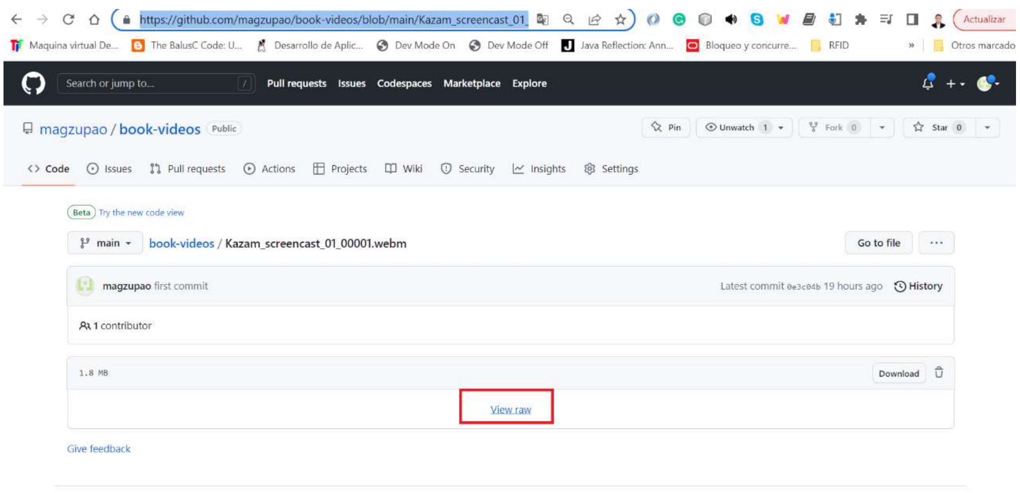
```
dev@debian:~/appsawtooth-pass$ docker ps -a
CONTAINER ID   IMAGE                                               COMMAND                  CREATED         STATUS                      PORTS
                           NAMES
43bede7e3ac1   hyperledger/sawtooth-shell:chime                    "bash -c 'sawtooth k…"   4 minutes ago   Up 4 minutes                4004/tcp, 8008/tcp
                           shell
7698475c7b10   hyperledger/sawtooth-devmode-engine-rust:chime      "devmode-engine-rust…"   4 minutes ago   Up 4 minutes
                           engine
208748d5af2a   hyperledger/sawtooth-settings-tp:chime              "settings-tp -vv -C …"   4 minutes ago   Up 4 minutes                4004/tcp
                           settings
8ced8e6ca29b   hyperledger/sawtooth-rest-api:chime                 "sawtooth-rest-api -…"   4 minutes ago   Up 4 minutes                4004/tcp, 0.0.0.0:8008->800
8/tcp, :::8008->8008/tcp   rest-api
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             validator


dev@debian:~/appsawtooth-pass$ 
```

The computer has deployed the resources defined in the architecture image:



https://github.com/magzupao/book-videos/blob/main/Kazam_screencast_01_00001.webm to download the video, click on 'View raw'

Online player https://webm.to/player/?lang=es if not viewable on the computer.



In the video, until the minute we visualize all the deployment of the Blockchain node, after the minute we visualize the status of all the services contained in the node, all okey.

In this web address you can have several examples of how to implement a Smart Contract that in Sawtooth language is known as 'Transaction'.

9. Develop Smart Contract

The functionality to be implemented in a Smart Contract will be: "**ticket sales**" for an event or service.

We list the basic functions to be developed:

- The user buys a ticket for the PERU vs GERMANY match.
- The user registers his purchase by entering his ID and the chosen SPORTS EVENT.
- The data is processed and the ticket is registered in the Blockchain chain.
- Only the user can check his ticket using his ID.

Development environment

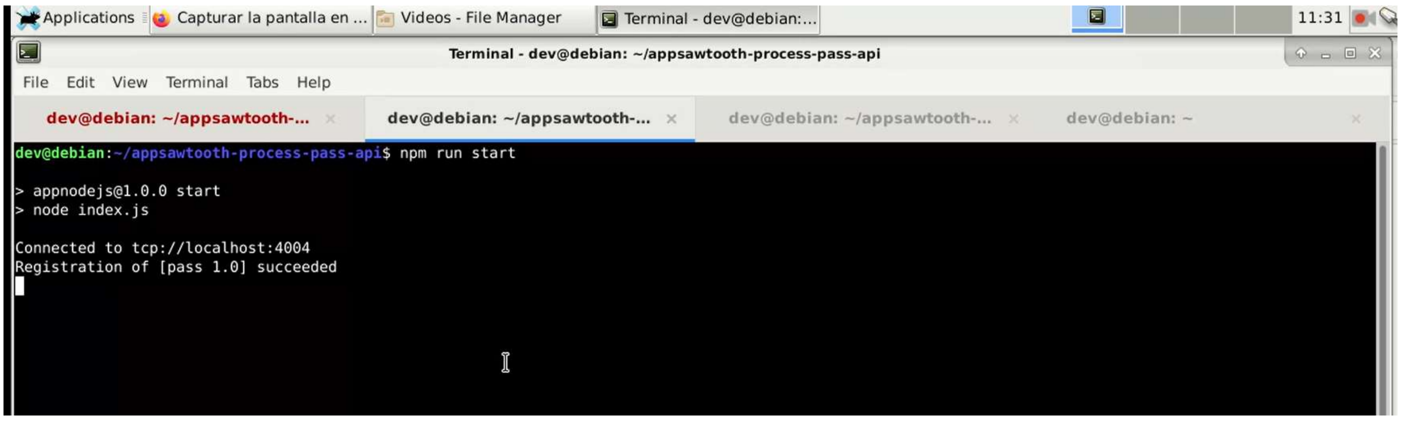
Our development environment has the following technical characteristics:

- Debian Linux v10, configuración básica (8 Mgbytes 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

Up to second 10 we visualize the Smart Contract loading, the contract is identified with the name of the contract:

pass: 1.0

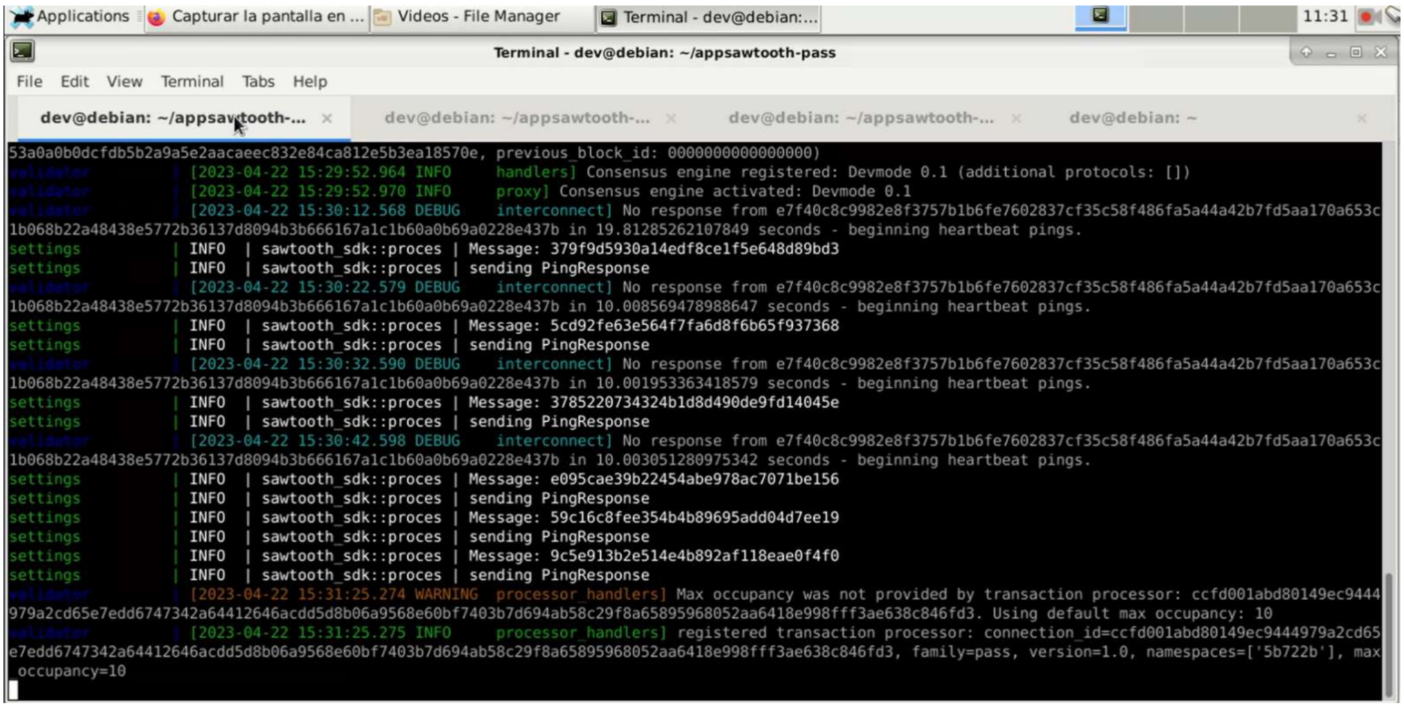


Applications
Capturar la pantalla en ...
Videos - File Manager
Terminal - dev@debian:...
11:31
Terminal - dev@debian: ~/appsawtooth-process-pass-api
File Edit View Terminal Tabs Help
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...
dev@debian: ~

```
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
```

Next, we visualize the contract record in the Blockchain node, as we see in the lower part of the image:

Family: pass, versión=1.0



Applications
Capturar la pantalla en ...
Videos - File Manager
Terminal - dev@debian:...
11:31
Terminal - dev@debian: ~/appsawtooth-pass
File Edit View Terminal Tabs Help
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...
dev@debian: ~

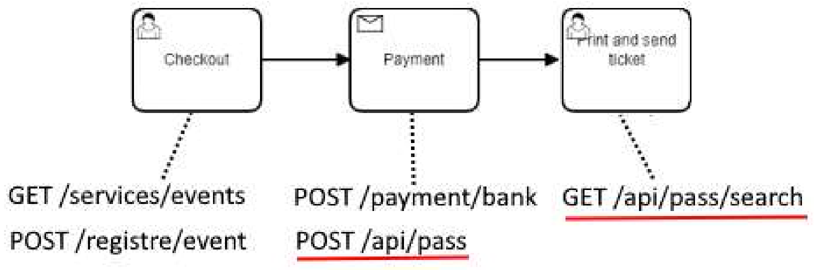
```
53a0a0b0dcfdb5b2a9a5e2aacaeec832e84ca812e5b3ea18570e, previous_block_id: 0000000000000000)
validator          | [2023-04-22 15:29:52.964 INFO     handlers] Consensus engine registered: Devmode 0.1 (additional protocols: [])
validator          | [2023-04-22 15:29:52.970 INFO     proxy] Consensus engine activated: Devmode 0.1
validator          | [2023-04-22 15:30:12.568 DEBUG    interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 19.81285262107849 seconds - beginning heartbeat pings.
settings           | INFO  | sawtooth_sdk::proces | Message: 379f9d5930a14edf8ce1f5e648d89bd3
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
validator          | [2023-04-22 15:30:22.579 DEBUG    interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.008569478988647 seconds - beginning heartbeat pings.
settings           | INFO  | sawtooth_sdk::proces | Message: 5cd92fe63e564f7fa6d8f6b65f937368
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
validator          | [2023-04-22 15:30:32.590 DEBUG    interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.001953363418579 seconds - beginning heartbeat pings.
settings           | INFO  | sawtooth_sdk::proces | Message: 3785220734324b1d8d490de9fd14045e
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
validator          | [2023-04-22 15:30:42.598 DEBUG    interconnect] No response from e7f40c8c9982e8f3757b1b6fe7602837cf35c58f486fa5a44a42b7fd5aa170a653c
1b068b22a48438e5772b36137d8094b3b666167a1c1b60a0b69a0228e437b in 10.003051280975342 seconds - beginning heartbeat pings.
settings           | INFO  | sawtooth_sdk::proces | Message: e095cae39b22454abe978ac7071be156
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
settings           | INFO  | sawtooth_sdk::proces | Message: 59c16c8fee354b4b89695add04d7ee19
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
settings           | INFO  | sawtooth_sdk::proces | Message: 9c5e913b2e514e4b892af118eae0f4f0
settings           | INFO  | sawtooth_sdk::proces | sending PingResponse
validator          | [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
validator          | [2023-04-22 15:31:25.275 INFO     processor_handlers] registered transaction processor: connection_id=ccfd001abd80149ec9444979a2cd65
e7edd6747342a64412646acdd5d8b06a9568e60bf7403b7d694ab58c29f8a65895968052aa6418e998fff3ae638c846fd3, family=pass, version=1.0, namespaces=['5b722b'], max
 occupancy=10
```

10. Using Smart Contract

To interact with the Smart Contract we need an interface, in this case it will be an API that can interact with the application. with the application.

The following flowchart represents an application or web service in production that is active and will integrate with the Blockchain node through the 'pass' API that exposes two services:

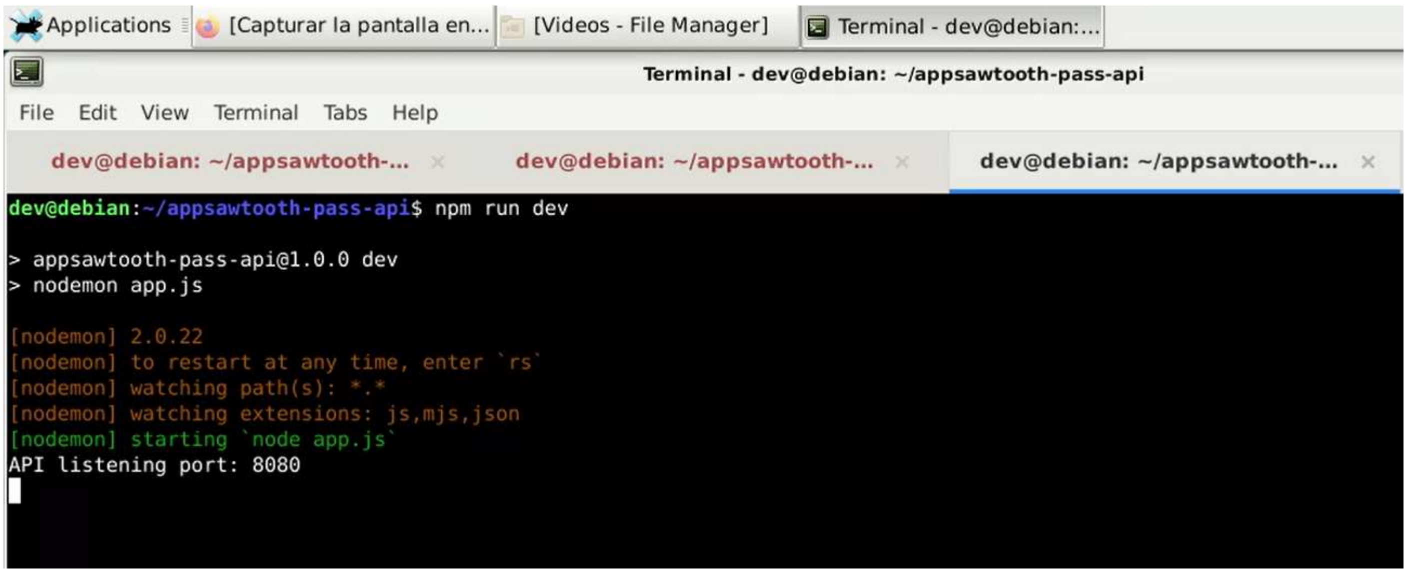
- POST we register the ticket in Blockchain.
- GET we query the ticket in Blockchain



WEB application in production that integrates with Blockchain through an API

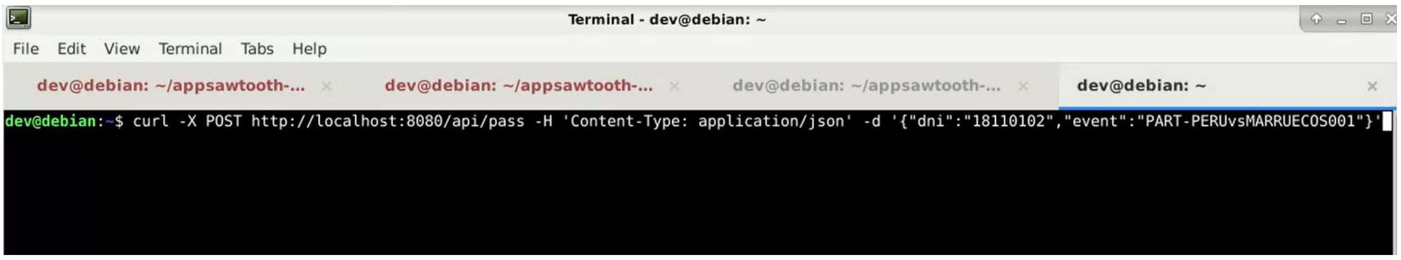
https://github.com/magzupao/book-videos/blob/main/Kazam_screencast_01_00003.webm

We initialize the API pass:


Applications
[Capturar la pantalla en...
[Videos - File Manager]
Terminal - dev@debian:...
Terminal - dev@debian: ~/appsawtooth-pass-api
File
Edit
View
Terminal
Tabs
Help
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...
dev@debian: ~/appsawtooth-...

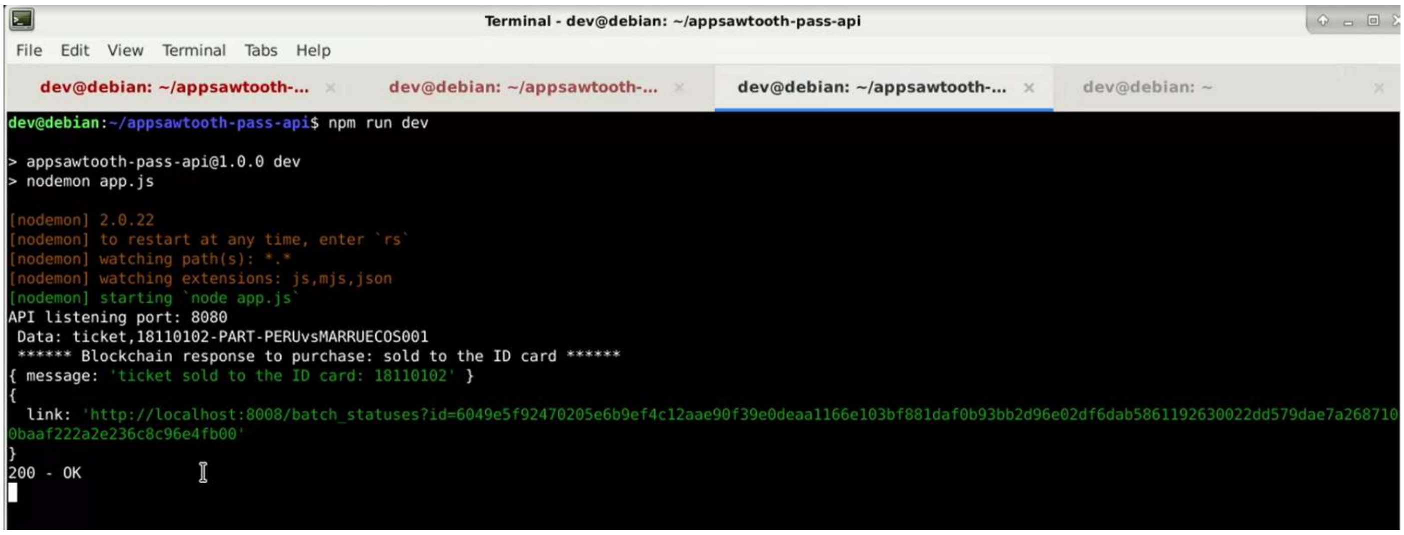
```
dev@debian:~/appsawtooth-pass-api$ npm run dev

> 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
```

We register an input - POST, in reality this call will be invoked from the web application.



```
dev@debian:~$ curl -X POST http://localhost:8080/api/pass -H 'Content-Type: application/json' -d '{"dni":"18110102","event":"PART-PERUvsMARRUECOS001"}'
```

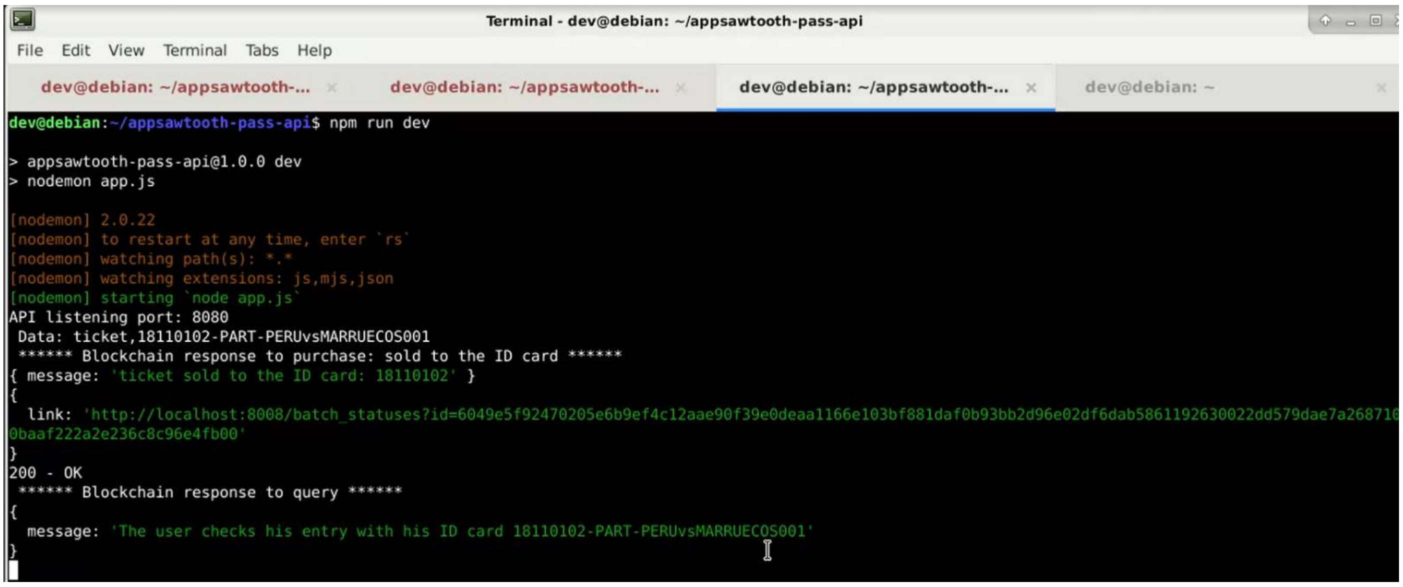
We get the answer from Blockchain:



```
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:8008/batch_statuses?id=6049e5f92470205e6b9ef4c12aae90f39e0deaa1166e103bf881daf0b93bb2d96e02df6dab5861192630022dd579dae7a268710
0baaf222a2e236c8c96e4fb00'
}
200 - OK
```

We consult the ticket created with GET:

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



```
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:8008/batch_statuses?id=6049e5f92470205e6b9ef4c12aae90f39e0deaa1166e103bf881daf0b93bb2d96e02df6dab5861192630022dd579dae7a268710
0baaf222a2e236c8c96e4fb00'
}
200 - OK
 ****** Blockchain response to query ******
{
  message: 'The user checks his entry with his ID card 18110102-PART-PERUvsMARRUECOS001'
}
```

Note - all the code used for this guide will be published in a single subscription where you will see everything in an easy and simple way. easy and simple. Saving you time and assimilating quickly all the concepts.

The next chapters will contain:

- Development of NFT's
- Designing and deploying an Enterprise Architecture

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=vcses
- 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://techdatatsmex.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