

UTILITIES

BLOCKCHAIN

Using the "Hyperledger Sawtooth" tool



2023

Mr. Marcus Aurelius Ford Zavaleta
Master in Corporate Software Development
<https://www.linkedin.com/in/marcoguado/>

INTRODUCCION

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. This technology was popularized by the cryptocurrency Bitcoin, 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 business 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 various programming languages and granular permission management for network users. In short, Hyperledger Sawtooth is a powerful and flexible Blockchain platform that can be tailored to the specific needs of a business.

This document will describe what is needed to implement a 'Smart Counter' such as:

- The installation of a Blockchain Node.
- Develop a 'Smart Contract'.
- Use the 'Smart Contract' through an API (Application Programming Interface).

INDEX

1. Definition
2. Decentralized database
3. Centralized database
4. Register data on the Blockchain
5. Blockchain services
6. Blockchain Platforms
7. Smart Contract
8. Configure development environment
9. Develop the Smart Contract
10. Use the Smart Contract.

1. Definition

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

2. Decentralized database

A decentralized database stores and distributes information (data) across multiple nodes or devices, rather than being 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 across the network to keep all copies synchronized.

The advantage of a decentralized database is that it offers greater resistance to failure and greater security, as there is no single point of failure or attack. In addition, a decentralized database can allow 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, 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, meaning there is no single point of failure or attack.
- **Redundancy:** Each node has a full copy of the data, which means that in the event that one 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.
- **Transparency:** Information can be accessed and updated transparently by all users 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 trust and transparency are required.
- **Synchronization:** Updates are propagated across the network to keep all copies of the data synchronized.
- **Scalability:** The decentralized database can scale efficiently as more nodes are added to the network.

In short, a decentralized database provides greater resilience to failure and increased security, and allows for 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 data over 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, as all data is in one place and security and privacy measures can be implemented in 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 and system failures.

In addition, in a centralized database, a central authority may be required to manage and control access to data, which can limit transparency and multi-user participation.

3.1. Characteristics

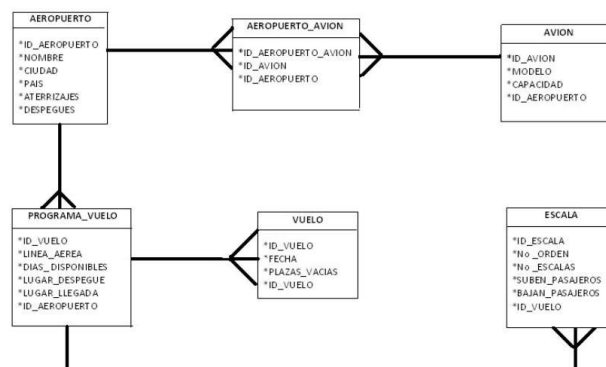
The characteristics of a centralized database are:

- Centralization: Information is stored on a single server or centralized location.
- Administration: Administration and maintenance of the database is done in one place, making it easy to manage.
- Access control: Access control to the database is performed on the central server, which allows to implement security and privacy measures effectively.
- Limited scalability: The centralized database has limited ability to scale as more data and users are added.
- Vulnerability to failure: A single point of failure can result in the loss of all information stored in the database.
- Network connection dependency: The network connection is essential for accessing the database, which can limit the availability of information if there are connection problems.
- Dependence on the central authority: The centralized database relies on a central authority to manage and control access to data, which can limit transparency and user participation.

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

Examples of centralized and decentralized databases

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

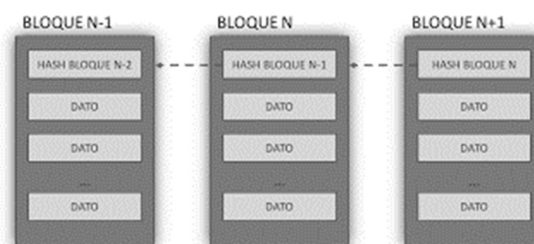


Copyright <http://basenatos.blogspot.com/2009/02/ejercicio-4-sistema-de-vuelos.html>

Under this model we can consult in the Database; what types of flights have a stopover, what flight usesn what type of plane, which airports have scheduled flights at night etc.

In summary, (centralized) databases store data under a scheme 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 it only stores extremely useful and shareable information.



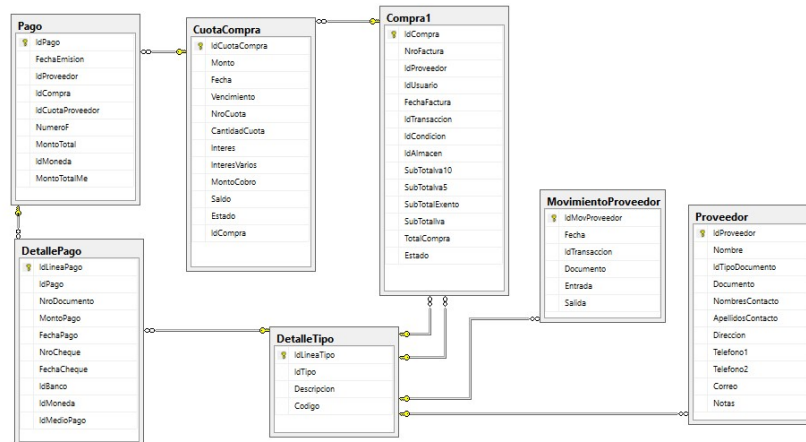
Copyright <https://www.aepd.es/es/prensa-y-comunicacion/blog/blockchain-ii-conceptos-basicos-proteccion-de-datos>

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

FACT = "Flight KL 744, AMS – LIM, pasj 234, fuel 20 tn, Jorge Chavez Airport"

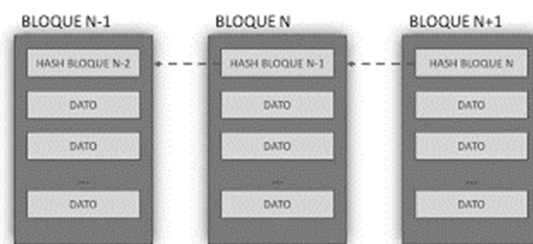
Blockchain is a distributed database that stores data differently, it is not in structure but in sequential.

Let's look at 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 of 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,NoQuota=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 we also 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 can not be copied or altered, the data to be saved would be something like this:

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

If the ticket is consumed, the **exchange** information would be recorded as follows:

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

With this we managed to disable the entrance.

"All these examples commented, are under the Blockchain service of type "Smart Contract" since they need to execute certain rules before saving the information"

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

4. Register data on the Blockchain

To register the data in Blockchain the user must do it by means of 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) database users have permissions to write and read, while in Blockchain it is written using the private certificate and read using the public certificate.

Every user who interacts with the Blockchain must do so using both its private and public certificates. Without the certificates, the information will not be accessible.

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 on the Blockchain. Certificates are usually physical files that have stored user data in encrypted form, these certificates are issued by those who manage the Blockchain infrastructure in which they interact.

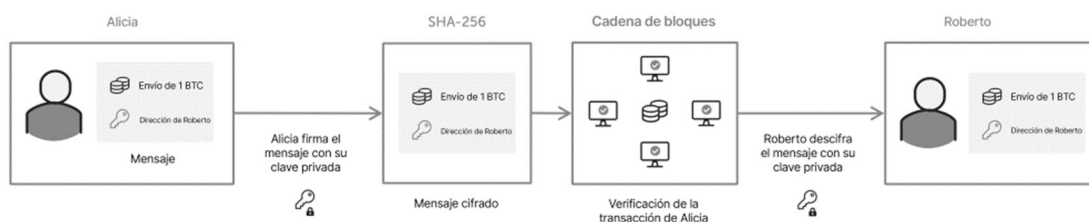
The following image graphs the usefulness of the certificates, although in the graph it represents cryptocurrencies, the procedure is the same for a data.

1. He sent a piece of information, for example, that of the ticket purchase:

Message => DATA = " PART-PERUvsGERMANY-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 Bob and only he using his public certificate (public key) can read the content of the information.



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

Note. - no hacker has broken the security of Blockchain, in all the news to date that have been published about the theft of Blockchain, hackers have seized the certificates (this being their Achilles heel) Blockchain.

But to correct this security point several strategies are followed, one is to delegate to each user the custody of their certificates or two to have a single user who validates all transactions.

5. Blockchain services

Blockchain technology can be useful to implement 3 services:

- Cryptocurrencies.
- Smart contracts known as: "Smart Contract"
- NFS tokens used 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 generator of cryptocurrencies, there are other services such as the "Smart Contract" which is what is used in most Blockchain systems when processing data, finally, we have the NFTs that together with the "Smart Contract" serve to ensure the authenticity and originality of some digital content. If we design a logo and distribute it on the Internet afterwards as we can claim copyright, this is achieved by joining NFT's + Smart Contract.

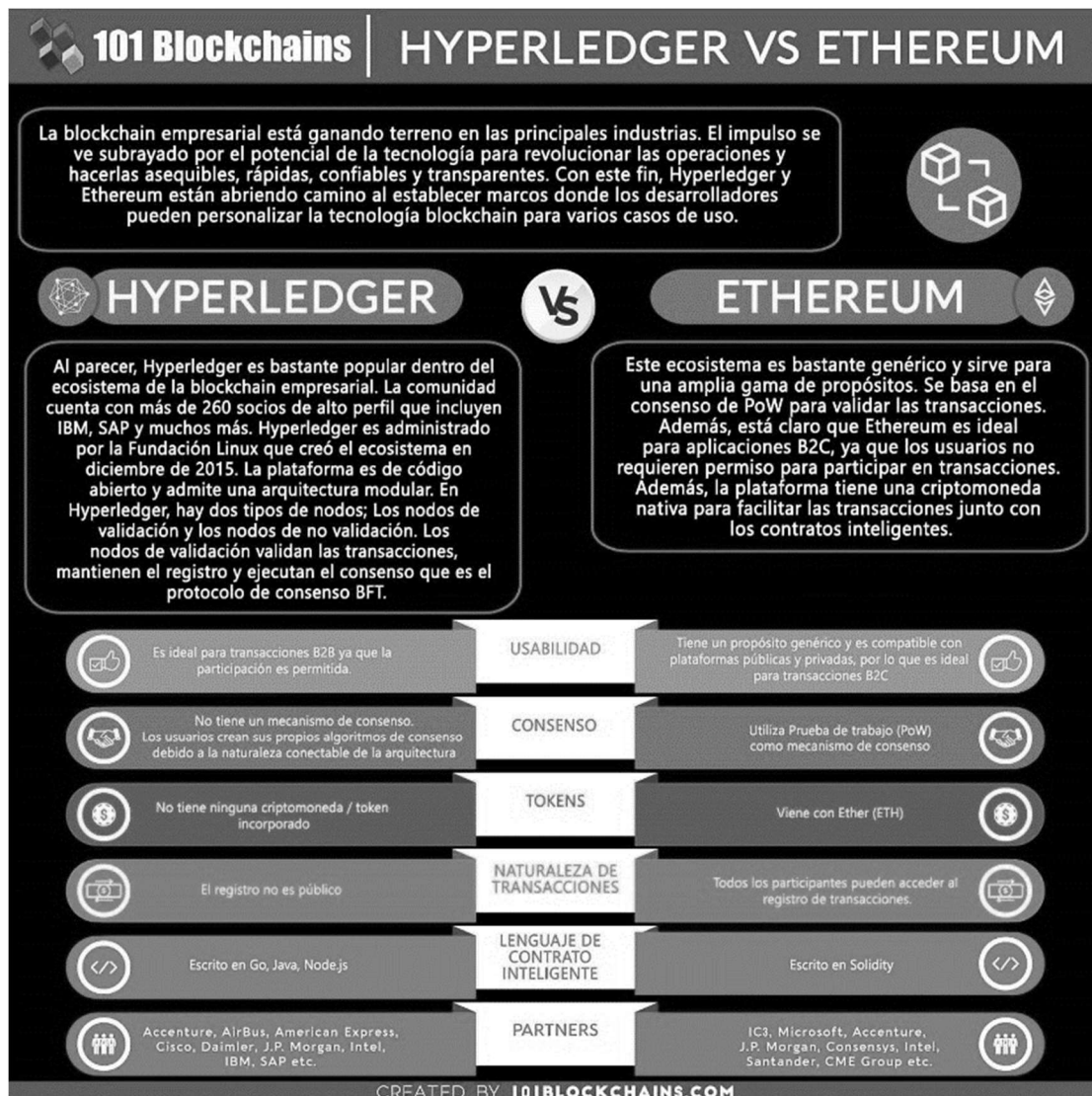
Another example is in audit issues where we can keep documents ensuring their authenticity and ensure that until that date they were not manipulated. In criminalistics using Blockchain technology is very useful to preserve the data of a test.

In summary: Blockchain technology is very useful when manipulating data and we want to ensure that those values will not be altered and we have the confidence to be able to share the information with 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 coin, instead 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 table between Hyperledger and Ethereum.



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

6. Blockchain Platforms

To program using Blockchain technology there are many platforms, in the following **graph** some are represented, 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://techdatasmex.blogspot.com/2018/03/las-5-mejores-plataformas-de-blockchain.html>

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

Note. – there is no need to use a currency to encrypt data on the Hyperledger Sawtooth Blockchain, if our business apart from encrypting data we want to generate a currency andntonces we can use Ethereum.

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

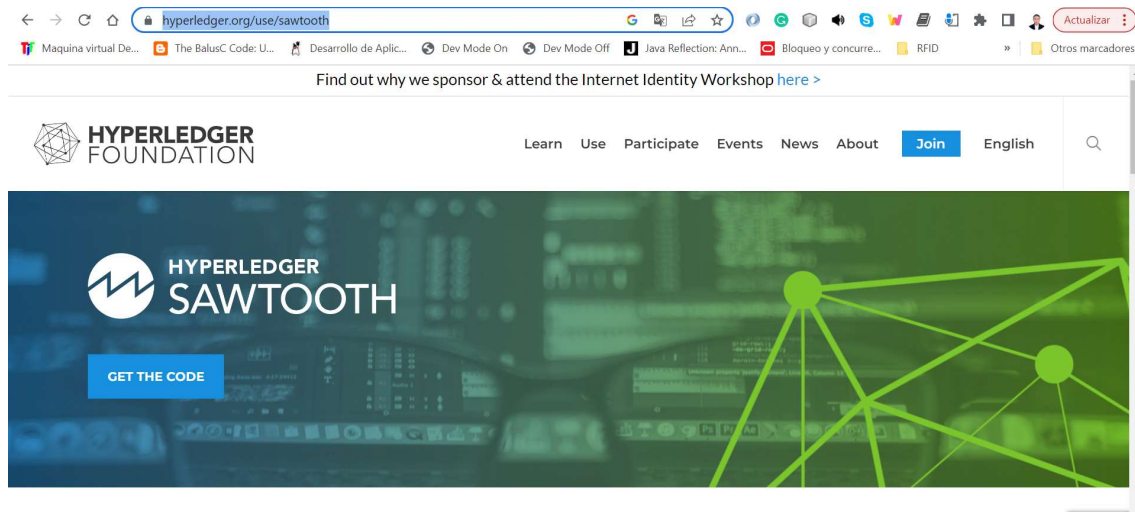


Copyright <https://codigoonlick.com/mejores-lenguajes-programacion-para-2018/>

To program a website, I must learn all those programming languages, of course not! Just choose a language.

The same goes for Blockchain, for thisguide we will 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. Sawtooth is designed to provide a scalable and flexible enterprise platform for Blockchain applications .

The platform is based on a modular architecture and enables the deployment of decentralized applications with a wide range of performance and privacy requirements. Sawtooth uses several consensus algorithms, 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 management, a web dashboard for network monitoring and management, and a RESTful API for integrating blockchain applications with other enterprise applications.

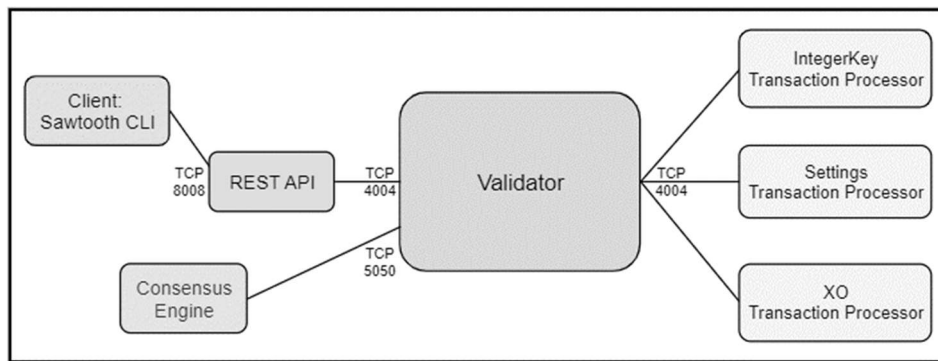
6.2. Features Hyperledger Sawtooth

Hyperledger Sawtooth is an open-source blockchain platform that is designed to be modular, scalable, and secure. Here are 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 based on the specific needs of the application.
- **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 into 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, making it easier for developers to build 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 simply by adding more nodes. In addition, Sawtooth uses a state partitioning model that allows for the parallelization of transactions, allowing the network to handle a higher volume of transactions.

In short, 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 the needs of their application. In addition, Sawtooth uses advanced security techniques and is designed to scale horizontally to handle a higher volume of transactions.

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 node?

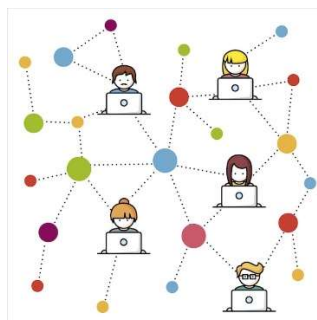
A Blockchain node is an essential component of a decentralized Blockchain network that helps maintain the integrity and security of the blockchain. A node can be defined as any device connected to the Blockchain network that participates in the process of validating transactions and confirming new blocks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

Each node in the network has a complete copy of the blockchain, which means that it contains all the data and transaction records since the creation of the chain. The nodes work together to validate transactions and confirm new blocks, which means 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 full copy of the blockchain and participate in transaction validation and new block creation, while light nodes only have a partial copy of the blockchain and rely on full nodes to verify transactions.

In short, 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. Differences of 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. 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 use in enterprise environments.

7. Smart Contract

Un Smart Contract or "smart contract" is a computer program that runs automatically on a Blockchain or blockchain. Essentially, a Smart Contract is a set of functionalities and conditions that are established and executed once the conditions programmed in 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 and digital identification systems. Because Smart Contracts are automatically executed in the Blockchain, they are very secure and cannot be altered once the conditions have been established.

"In practice, a Smart Contract is a function that is programmed, responds to a data entry to execute an action or returns a response – data"

A Smart Contract is a set of functionalities that are programmed and executed automatically on the Blockchain. These functionalities define the conditions that must be met for the contract to be executed and establish the logic of the transaction, 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 conditions of the contract are established, they cannot be altered, which guarantees the integrity of the process.

In summary, Smart Contracts are computer programs that establish functionalities and conditions, and are automatically executed in 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 knowledge in programming and a basic understanding of how a Blockchain works. To program a Smart Contract we must perform these 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. For example, Ethereum is one of the

most used Blockchain platforms for the creation of Smart Contracts, but there are other platforms 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 creating Smart Contracts. For example, Ethereum uses Solidity, while Hyperledger Sawtooth uses Go, Java or JavaScript. Make sure you choose the right programming language for the Blockchain platform you've 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. This involves determining the conditions that must be met for the contract to be executed and establishing the logic of the transaction, determining the input and output of data.
4. **Write the code:** With the Smart Contract rules defined, it's time to write the code. Use the selected programming language to write the Smart Contract code, making sure to follow programming best practices and comply with Blockchain platform standards.
5. **Test the Smart Contract:** Once you have written the code of the Smart Contract, it is important that you 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 deployed 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 platform.
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 app's needs change.

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

7.2. Programar un Smart Contract

To create a Smart Contract program using Hyperledger Sawtooth, we do the following :

1. **Set up the development environment:** This involves installing the required packages, setting up the runtime environment, and setting up the test environment. For more information on setting up the Sawtooth development environment, see the official documentation.
2. **Programming the functionalities of the Smart Contract:** this involves defining the programming logic of the functionalities that the contract must comply with as well as validating the input and output of data.
3. **Select the programming language:** Sawtooth supports several programming languages, including 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 you 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 correct errors before the contract is deployed on the network.
6. **Deploy the Smart Contract:** Once you've tested the Smart Contract and are sure it's working properly, it's time to install it into 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 app's needs change.

In summary, to create a Smart Contract using Hyperledger Sawtooth, you must configure 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 to production, monitor and update the contract as needed.

8.Configure development environment

8.1 configure the Blockchain node

Setting up a Hyperledger Blockchain 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 Linux -v10 operating system with these technical characteristics:

- RAM 8 Mbytes
- 64 Mbytes hard drive
- 4 colors

The installed software is as follows:

- Git
- Docker
- Docker-compose
- NodeJs

The Hyperledger Sawtooth nodes are going to be instantiated using the [docker-compose](#) tool, in the following image we visualize the instantiated nodes:

```

[0] Appl[Applications] Terminal - dev@debian: ... 05:53
Terminal - dev@debian: ~ - appswatooth-pass
File Edit View Terminal Tabs Help

dev@debian: ~ - appswatooth-pass x dev@debian: ~ - appswatooth-pass

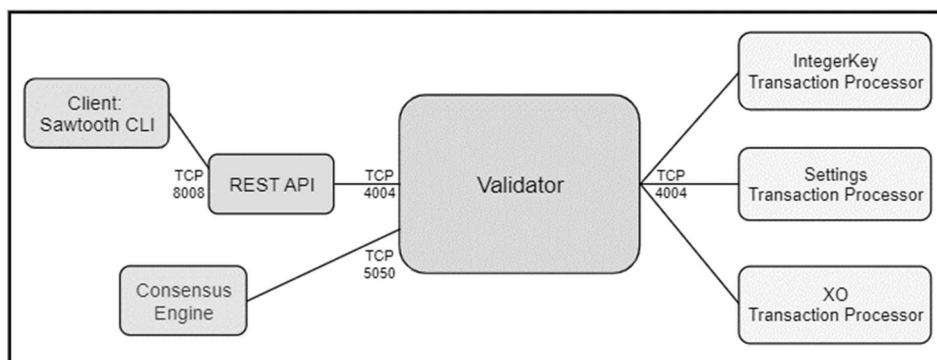
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.92176840977 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0606195762024 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 3954226307b63047b9c239034f0ed72
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0303817953491 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 68818a6b74c3097b63c0b3140474
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] settings INFO | swatooth sdk::process | Message: 8b30f874638fc431fa1a00a33d410043
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] settings INFO | swatooth sdk::process | Message: 43186d9a264c70903494e1273f031
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] settings INFO | swatooth sdk::process | Message: e05f5e2b49484eadd12defc72a757
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0746935871582 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 68418a6b74c3097b63c0b3140474
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] settings INFO | swatooth sdk::process | Message: 8b30f874638fc431fa1a00a33d410043
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] settings INFO | swatooth sdk::process | Message: 43186d9a264c70903494e1273f031
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0804968493189 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 6e474ea3f760b4a472bc3b18c355
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0804968493189 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 61583193373ad0810430d9e0e7512
[0] settings INFO | swatooth sdk::process | sending PingResponse
[0] d/535919a3580e2e97b1b84a5c8082a7f17ca3d0c74f15514 in 19.0804968493189 seconds - beginning heartbeat pings.
[0] settings INFO | swatooth sdk::process | Message: 6c724c2547f44949a32d47c14370b7f6
[0] settings INFO | swatooth sdk::process | sending PingResponse

```


We visualize the instantiated nodes and check the status, they are all 'UP' – okay.

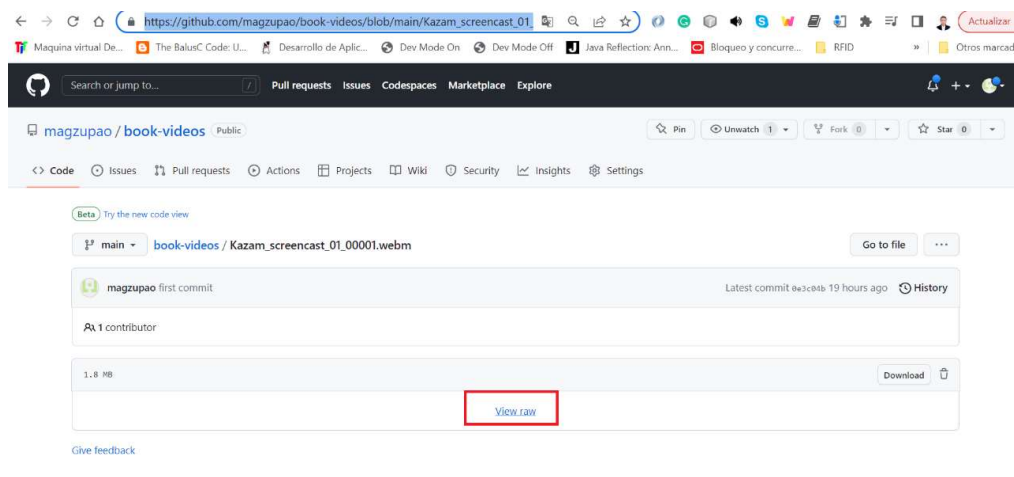
```
Applications Terminal - dev@debian: ... 05:54 dev
Terminal - dev@debian: ~/appsawtooth-pass
File Edit View Terminal Tabs Help
dev@debian: ~/appsawtooth-pass dev@debian: ~/appsawtooth-pass
dev@debian:~/appsawtooth-pass$ docker ps -a
CONTAINER ID   IMAGE                                     COMMAND                  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$
```

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



https://github.com/magzupao/book-videos/blob/main/Kazam_screencast_01_00001.webm to download the video you have to give 'View raw'.

Online player <https://webm.to/player/?lang=es> if it cannot be viewed on the computer.



In the video, up to the minute the entire deployment of the Blockchain node is displayed, after the minute we visualize the status of all the services contained in the node, all okay.

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

9 Develop the 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 match PERU vs GERMANY
- The user registers his purchase by entering his ID and the chosen SPORTING EVENT
- The data is processed and the input is recorded the Blockchain chain.
- Only the user can consult his entry using his ID

Development environment

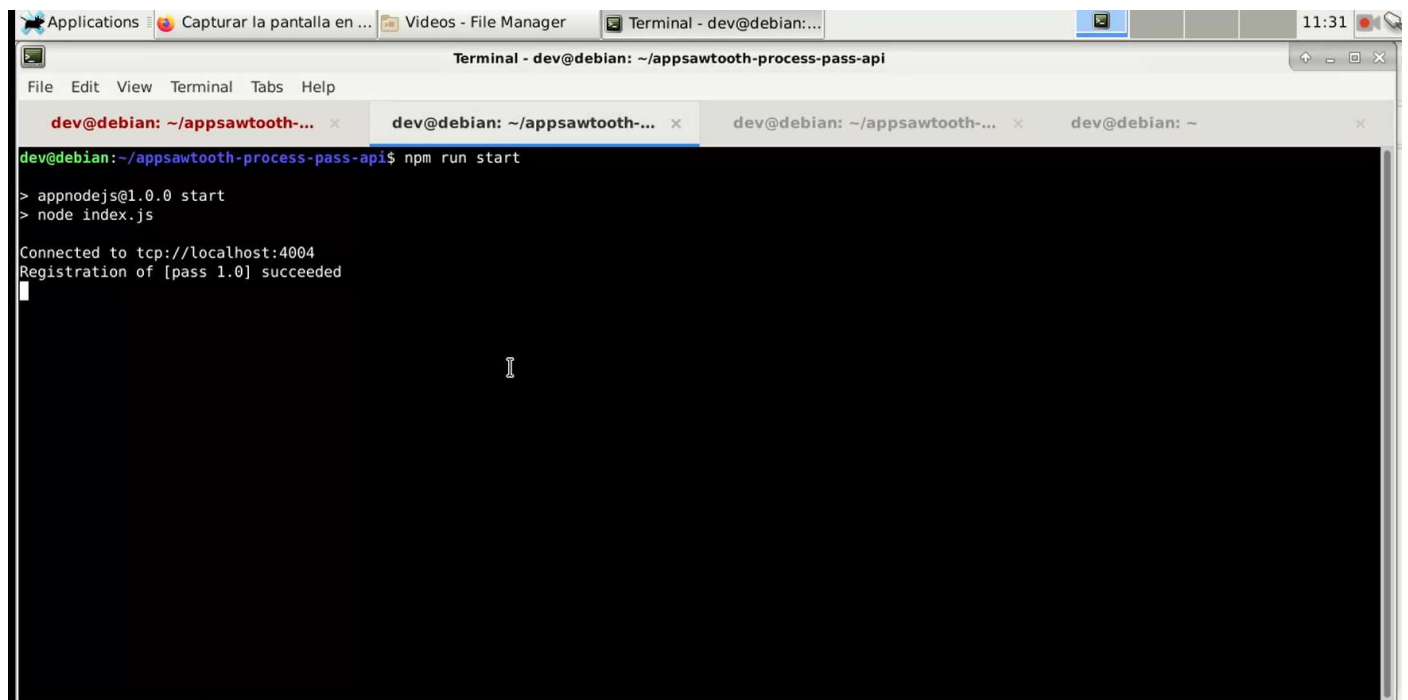
Our development environment has the following technical characteristics:

- Debian Linux v10 operating system, basic configuration (8 Mbytes RAM)
- IDE VScode
- NodeJs v16.19.1
- Npm v8.19.3

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

Until the second 10 we visualize the load of the Smart Contract, the contract is identified with the name:

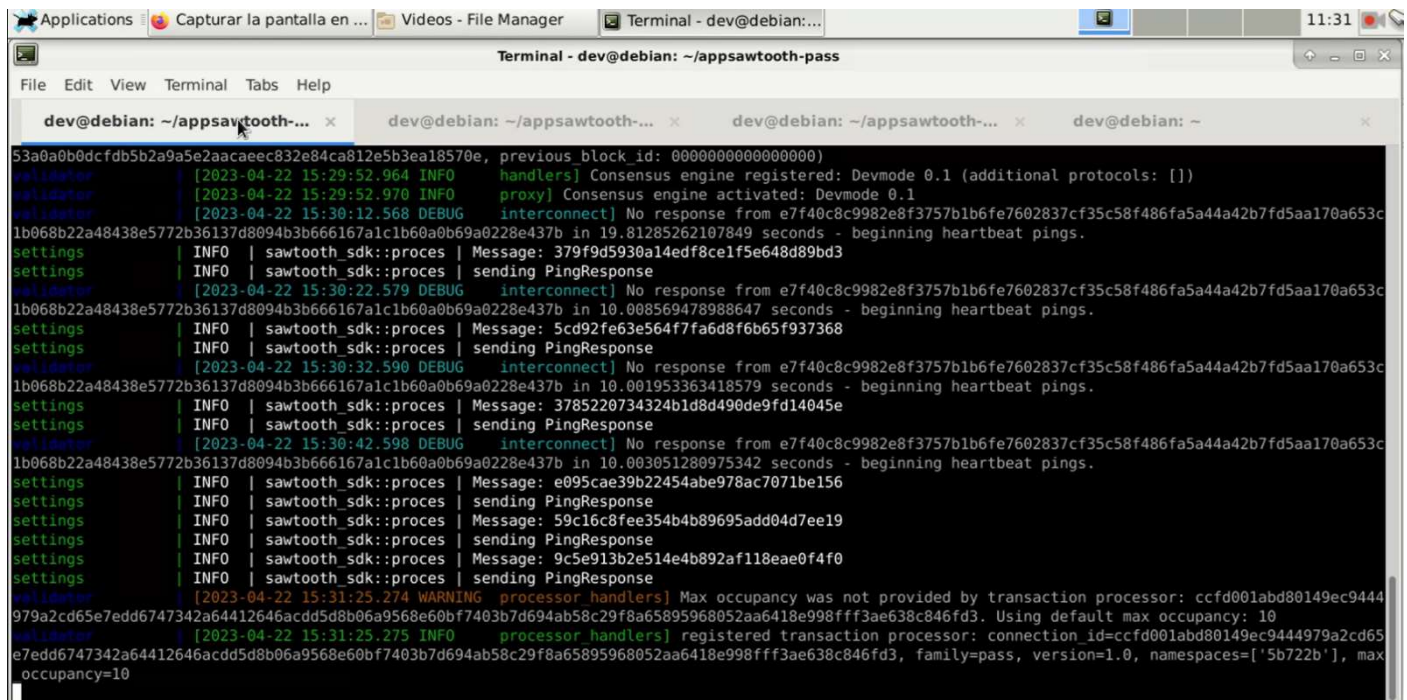
pass: 1.0



```
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
```

Then, we visualize the contract record in the Blockchain node, as we see at the bottom of the image:

Family: pass, version=1.0



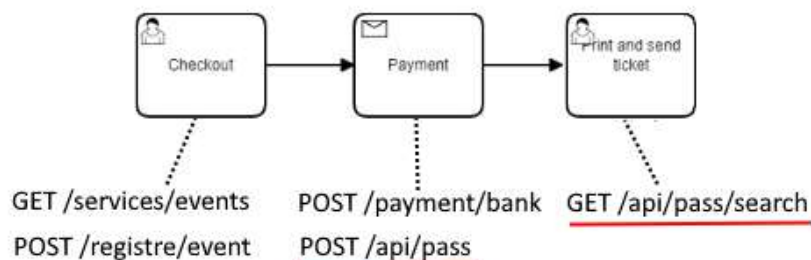
10. Use the Smart Contract

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

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

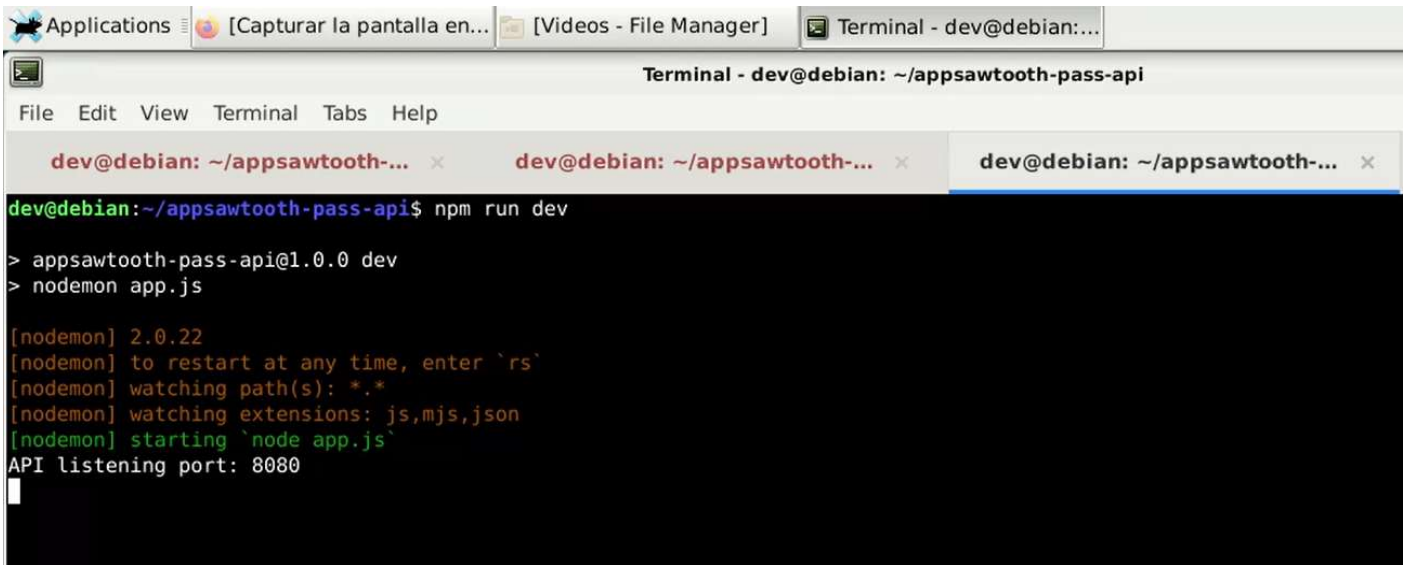
POST we register the ticket in Blockchain

GET we consult the ticket in Blockchain



WEB application in production that integrates with Blockchain through an API

Initialize the API pass:



A terminal window titled "Terminal - dev@debian: ~/appsawtooth-pass-api" with tabs for "Applications", "[Capturar la pantalla en...", "[Videos - File Manager]", and "Terminal - dev@debian:...". The terminal shows the following commands and output:

```
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
```

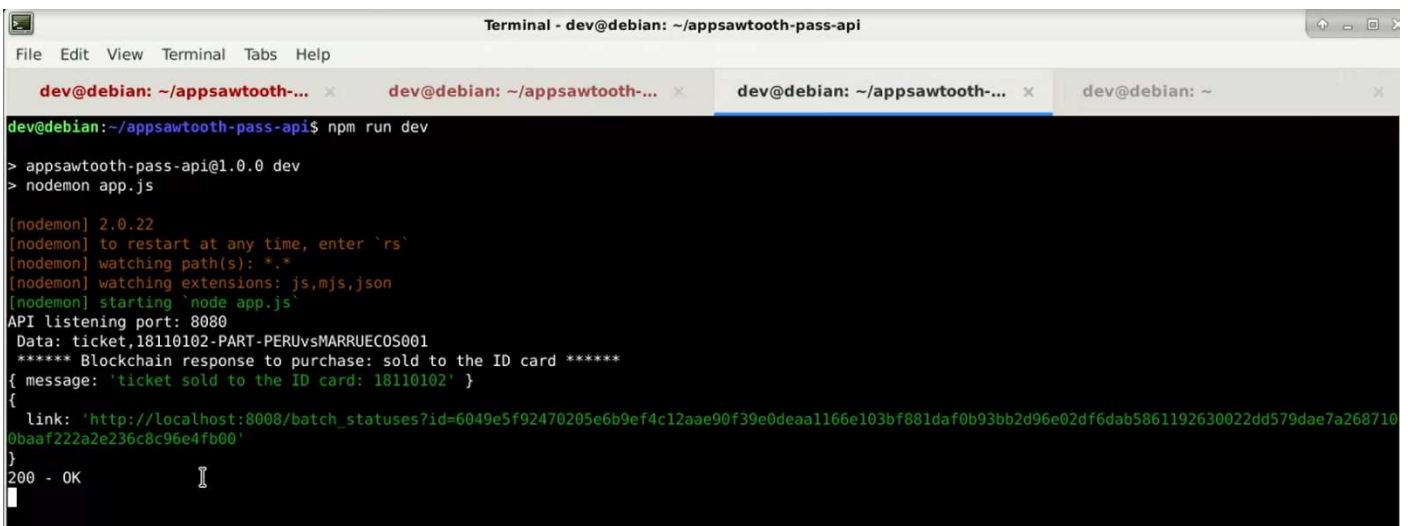
We register an entry – POST, in reality this call will be invoked from the web application.



A terminal window titled "Terminal - dev@debian: ~" with tabs for "dev@debian: ~/appsawtooth-...", "dev@debian: ~/appsawtooth-...", "dev@debian: ~/appsawtooth-...", and "dev@debian: ~". The terminal shows the following command and output:

```
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:



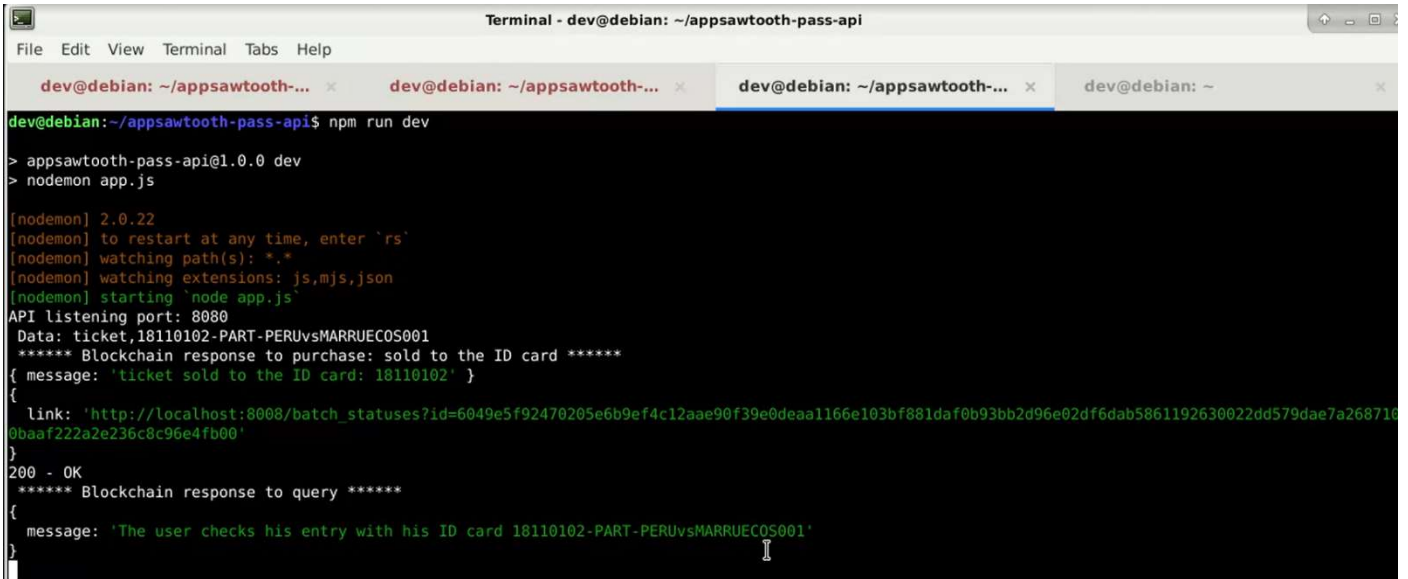
A terminal window titled "Terminal - dev@debian: ~/appsawtooth-pass-api" with tabs for "dev@debian: ~/appsawtooth-...", "dev@debian: ~/appsawtooth-...", "dev@debian: ~/appsawtooth-...", and "dev@debian: ~". The terminal shows the following commands and output:

```
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
```

We consult the ticket created with 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'
}
```

Note. - All the code used for this guide will be published in a single subscription where you will see everything easily and simply. Lo that saves you time and quickly assimilate all concepts.

The next chapters will contain:

- NFT's Development
- Design and deployment of an Enterprise architecture

RESOURCES

- <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