Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Centro de Investigaciones en Comunicaciones y Redes (CICORE)



Solución de Monitoreo de Cadena de Suministro con Contrato Inteligente

Trabajo especial de grado

presentado ante la Ilustre

Universidad Central de Venezuela

Por el bachiller:

Gianfranco Manfredi

C.I.N° 25.217.607

[franco.manfredi2006@gmail.com](mailto:franco.manfredi2006@gmail.com)

Tutor:

Prof. Antonio Russoniello

Caracas,

Agradecimientos

Resumen

**Título:**

Solución de monitoreo de cadena de suministros con contrato inteligente

**Autor:**

Br. Gianfranco Manfredi

**Tutor:**

Prof. Antonio Russoniello

**Resumen:**

Índice General

[Introducción 8](#_Toc147698502)

[1.Plantamiento del problema 9](#_Toc147698503)

[1.1 Justificación 10](#_Toc147698504)

[1.2 Objetivo general 10](#_Toc147698505)

[1.3 Objetivos específicos 10](#_Toc147698506)

[1.4 Alcance 10](#_Toc147698507)

[2. Marco conceptual 10](#_Toc147698508)

[2.1 Blockchain (Cadena de bloque) 10](#_Toc147698509)

[2.1.1 Estructura 11](#_Toc147698510)

[2.1.2 Redes peer-to-peer (par-a-par) 11](#_Toc147698511)

[2.1.3 Bloque 12](#_Toc147698512)

[2.1.4 Funcionamiento 15](#_Toc147698513)

[2.1.5 Características 16](#_Toc147698514)

[2.1.5.1 Descentralización 16](#_Toc147698515)

[2.1.5.2 Transparencia 18](#_Toc147698516)

[2.1.5.3 Inmutabilidad 18](#_Toc147698517)

[2.1.6 Aplicaciones 19](#_Toc147698518)

[2.1.7 Fork (Bifurcación) 20](#_Toc147698519)

[2.1.8 Tipos de Blockchain 20](#_Toc147698520)

[2.2 Wallets (Billeteras digitales) 21](#_Toc147698521)

[2.2.1 Transferir Fondos 21](#_Toc147698522)

[2.2.2 Tipos 21](#_Toc147698523)

[2.2.3 Almacenamiento en frio 22](#_Toc147698524)

[2.2.4 Almacenamiento en caliente 22](#_Toc147698525)

[2.2.5 mecanismos de consenso 23](#_Toc147698526)

[2.2.5.1 Proof-of-work (PoW) 23](#_Toc147698527)

[2.2.5.2 Proof-of-stake (PoS) 24](#_Toc147698528)

[2.2.5.3 Proof-of-Burn (PoB) 25](#_Toc147698529)

[2.2.5.4 Delegate-Proof-of-Stake (DPoS) 25](#_Toc147698530)

[2.2.5.5 Proof-of-Capacity (PoC) 26](#_Toc147698531)

[2.2.5.6 Comparación 26](#_Toc147698532)

[2.3 Etherium 27](#_Toc147698533)

[2.3.1 Comparación con Bitcoin 27](#_Toc147698534)

[2.3.2 Ventajas y desventajas 28](#_Toc147698535)

[2.3.3 Ethereum y aplicaciones Descentralizadas DApps 28](#_Toc147698536)

[2.3.4 Nodos Etherium 28](#_Toc147698537)

[2.3.5 Moneda de Etherium 29](#_Toc147698538)

[2.4 Contratos inteligentes (Smart Contracts) 30](#_Toc147698539)

[2.4.1 Lenguajes 30](#_Toc147698540)

[2.4.2 Solidity 31](#_Toc147698541)

[2.4.3 Herramientas de desarrollo 32](#_Toc147698542)

[2.4.4 Entornos de desarrollo 33](#_Toc147698543)

[2.4.5 Marcos de desarrollo (Frameworks) 33](#_Toc147698544)

[2.4.6 Librerias y clientes 34](#_Toc147698545)

[2.4.7 API de Infura 34](#_Toc147698546)

[2.5 Almacenamiento de datos 36](#_Toc147698547)

[2.5.1 IPFS (Sistema de archivos interplanetario) 36](#_Toc147698548)

[2.5.1.1 Componentes 38](#_Toc147698549)

[2.5.2 Swarm 39](#_Toc147698550)

[2.5.2.1 Componentes 41](#_Toc147698551)

[2.6 Placas de desarrollo 41](#_Toc147698552)

[2.6.1 Heltec-ESP32 42](#_Toc147698553)

[2.6.2 DHT-11 43](#_Toc147698554)

[2.6.3 Adafruit GPS 45](#_Toc147698555)

[2.7 Redes IoT 46](#_Toc147698556)

[2.7.1 LoRaWAN 47](#_Toc147698557)

[2.7.2 Sigfox 47](#_Toc147698558)

[2.8 APIs de geolocalización 49](#_Toc147698559)

[2.8.1 Google Maps 49](#_Toc147698560)

[2.8.2 Open Street map 49](#_Toc147698561)

[2.9 Trabajos relacionados 50](#_Toc147698562)

[2.9.1 MasterCard Blockchain 50](#_Toc147698563)

[2.9.1 IBM FoodTrust 50](#_Toc147698564)

[3.Marco Aplicativo 52](#_Toc147698565)

[3.1. Metodología General de Trabajo 52](#_Toc147698566)

[3.1.1 SCRUM 52](#_Toc147698567)

[3.1.1.1 Roles principales de SCRUM 52](#_Toc147698568)

[3.1.1.2. Time Boxes de SCRUM 53](#_Toc147698569)

[3.1.1.3 Artefactos 53](#_Toc147698570)

[3.2. Modificaciones al entorno de trabajo SCRUM 54](#_Toc147698571)

[3.3 Tecnologías utilizadas 55](#_Toc147698572)

[4. Diseño de la aplicación 56](#_Toc147698573)

**Índice de Figuras**

[Figura 2.1.2.1: Red peer-to-peer 13](#_Toc147698574)

[Figura 2.1.3.2: Composición de una cadena de bloque 13](#_Toc147698575)

[Figura 2.1.3.3 Cabezal de un bloque 14](#_Toc147698576)

[Figura 2.1.5.4: cadena de bloque Ethereum 18](#_Toc147698577)

[Figura 2.1.5.4: Ejemplo de función hash de SHA256. 19](#_Toc147698578)

[Figura 2.4.2.5: Ejemplo de código de solidity 32](#_Toc147698579)

[Figura 2.4.7.6: Enlace de Infura [12]. 35](#_Toc147698580)

[Figura 2.5.1.7: Funcionamiento de IPFS [13] 37](#_Toc147698581)

[Figura 2.6.1.8: dispositivo Heltec-ESP32[16] 42](#_Toc147698582)

[Figura 2.6.2.9: Trama de datos del DHT-11 43](#_Toc147698583)

[Figura 2.6.2.10: Dispositivo DHT-11[17] 44](#_Toc147698584)

[Figura 2.6.3.11: Dispositivo Adafruit GPS[18] 45](#_Toc147698585)

[Figura 2.7.2.12: Sistema en la nube de sigfox [19] 48](#_Toc147698586)

[Figura 3.1.1.1: Estructura completa de un sprint en SCRUM [16] 54](#_Toc147698587)

**Índice de Tablas**

[Tabla 2.1.2.1: Campos del cabezal de un bloque. 14](#_Toc147698265)

[Tabla 2.2.5.6.2: comparación de mecanismos de consenso. 26](#_Toc147698266)

[Tabla 2.3.5.3: Conversión de la moneda de Ethereum 29](#_Toc147698267)

[Tabla 3.2.1: tabla de sprints. 54](#_Toc147698268)

# Introducción

Una de las tendencias que han tenido un gran auge e impacto en el mundo de la tecnología ha sido blockchain, que rompió con el paradigma de una red de información centralizada a uno descentralizado, donde la gente puede ser dueña de los datos que almacena y comparte, creando así un ambiente de confianza entre usuarios que pertenecen a una red descentralizada para crear sus propias ideas en esta nueva plataforma.

Con la llegada del Bitcoin, se puedo contar con la primera implementación de blockchain, donde cada usuario podía crear cualquier solución que utilizara blockchain dado a su naturaleza de código abierto, generando así el nacimiento de otro tipo de criptomonedas derivadas de Bitcoin.

Ethereum fue desarrollado como una de estas derivaciones, que se diferencia totalmente de Bitcoin, su creador Vitalik Buterin tuvo la idea de volver la cadena de Ethereum una súper computadora utilizando los nodos de la misma para poder desarrollar aplicaciones de manera descentralizada, naciendo así los contratos inteligentes que son usados en conjunto con otras herramientas para hacer aplicaciones a gran o pequeña escala, utilizando el poder de la blockchain de Ethereum. Siguiendo este orden de ideas, este documento surge para plantear una solución, basada en la tecnología de contrato inteligente, que permita registrar y procesar información del ambiente a que está sometida una carga de productos. El documento se encuentra organizado como se describe a continuación:

* Capítulo 1: Planteamiento del problema a solventar, basado en problemas de logística y confiabilidad que poseen las cadenas de suministro hoy en día en Venezuela.
* Capítulo 2: Marco conceptual, se definen conceptos generales de la blockchain, carteras electrónicas, mecanismos de consenso, cómo funciona la cadena Ethereum, explicación del funcionamiento de IPFS, ejemplo de uso de dispositivos IoT y Api de geolocalización.
* Capítulo 3: Se establece un Marco Aplicativo, en el que se presentan en detalle las tecnologías y la metodología de trabajo utilizadas.
* Capítulo 4: Diseño e implementación de la solución, en el que se expone el diseño del sistema y su implementación de acuerdo con la metodología descrita en el capítulo 3 de este trabajo de investigación.
* Capítulo 5: Contribución, Limitaciones y Trabajos futuros.
* Conclusiones, referencias bibliográficas y diferentes anexos relacionados a la solución.

# 1.Plantamiento del problema

La cadena de suministros es uno de los elementos claves para la logística de cualquier empresa, asegurar una buena planificación y un buen funcionamiento de esta conlleva a la satisfacción del cliente y al éxito de la empresa que ofrece el servicio, ya que la importancia de la cadena radica en que engloba aquellas actividades asociadas con el movimiento de bienes desde el suministro de materias primas, la fabricación y la distribución hasta el usuario final. Pero cuando los productos son transportados del almacén al cliente final pueden ser sometidos a factores externos que alterarían la integridad del producto causando que lleguen en mal estado o que ni siquiera llegue al cliente final lo que causaría costos adicionales por reposición de mercancía. Algunos escenarios de factores que podrían afectar el suministro de producto son:

* Una empresa de alimentos no perecederos o de medicamentos transporta una carga que requiere estar sometida a una temperatura baja y, por un mal funcionamiento, el equipo de refrigeración del transporte falla en el camino haciendo que el producto se descomponga o dañe.
* Los transportistas por motivos personales estén descontentos con la empresa y en plena ruta de transporte procedan a robarse la mercancía.
* Mal sellado del contenedor en donde son transportado los alimentos aumente la humedad lo que incrementa la propagación de hongos en los productos generando un problema de salud pública.

Por más que las empresas planifiquen el transporte de sus productos no pueden prevenir estos factores externos porque escapan de su control y no tienen las herramientas necesarias para prevenir este tipo de problemas.

## Justificación

Considerando que las empresas no poseen una infraestructura para el monitoreo de la cadena de suministro de sus productos, se plantea en este trabajo de investigación, el desarrollo de una solución de monitoreo basado en contrato inteligente en conjunto con los dispositivos IoT (Hectec,DHT-11, Adafruit GPS) y una interfaz web compatible con contrato inteligente registre datos de GPS, humedad del aire y temperatura generando así alertas que les permita a los clientes monitorear de manera fiable una ruta de suministros para prevenir pérdidas o minimizar daños de mercancía y asegurar el éxito del envío.

## Objetivo general

Desarrollar una solución basada en contrato inteligente que nos permita visualizar la ruta de una de suministros y registrar las variables ambientales alas que está sometida una carga.

## Objetivos específicos

1. Definir una arquitectura base de hardware y software para el desarrollo de la plataforma.
2. Desarrollar contrato inteligente para integración con IPFS y la interfaz web.
3. Desarrollar un motor API-RESTFUL para las inserciones de datos de los sensores y la integración con IPFS y el contrato inteligente en la blockchain de Ethereum.
4. Desarrollo de interfaz web para integración entre IPFS y contrato inteligente.
5. Integración de mapas con OpenStreetMap en la interfaz web para visualización de rutas y valores de las variables ambientales.
6. Realizar las pruebas pertinentes para asegurar la integración y la calidad de la plataforma.

## Alcance

El alcance de esta propuesta contempla el desarrollo de un contrato inteligente y simulación sensores para registrar datos de GPS, humedad del aire y temperatura que son almacenados en IPFS y a su vez son registrados en la blockchain y por medio de este procesar los datos para poder alimentar una interfaz web de monitoreo para que genere alertas con los datos suministrados y poder visualizarlo en una interfaz amigable para el usuario.

# 2. Marco conceptual

En este capítulo se exponen y definen diversos conceptos y la investigación realizada sobre las tecnologías relacionadas que permitirán abordar el problema y el diseño de su solución de una forma estructurada.

## 2.1 Blockchain (Cadena de bloque)

La blockchain (cadena de bloque) es una serie de registros inmutables de datos que están manejados por un cluster (conglomerado) de computadoras sin pertenecerle a una sola entidad [1].

La red de blockchain (cadena de bloque) no tiene una autoridad central. Como es un conjunto de capas inmutables compartidas, la información está abierta para todos aquellos que pertenezcan a esa cadena.

### 2.1.1 Estructura

Una cadena de cualquier criptomoneda está compuesta por un conglomerado de nodos (bloques) interconectados por una red peer-to-peer (par-a-par) donde cada uno de los bloques se rige por unas reglas (protocolos) establecido por la comunidad para crear un consenso entre todos los bloques. Los bloques son computadoras individuales que toman la entrada y realizan una función en ellas y dan una salida. La carga de trabajo se reparte entre todos los participantes, que son igualmente privilegiados.

Cada blockchain tiene una estructura distinta pues, al momento de crear una criptomoneda la comunidad responsable puede establecer las reglas que sean convenientes para poder transar con esa criptomoneda, si tomamos de ejemplo Bitcoin y Ethereum tenemos que el bloque de Bitcoin es de un tamaño fijo de 1MB por bloque en comparación con el bloque de Ethereum que no tiene un tamaño definido.

### 2.1.2 Redes peer-to-peer (par-a-par)

Es una red de nodos interconectados entre sí que actúan simultáneamente como cliente-servidor. Uno de los principales usos de la red peer-to-peer es el intercambio de archivos conocido como torrenting. Si se va a utilizar un modelo cliente- servidor para la descarga, generalmente es extremadamente lento y depende completamente de la salud del servidor, además, es propenso a la censura [1].

Sin embargo, en un sistema peer-to-peer no hay una autoridad central, por lo tanto, incluso si uno de los pares de la red se sale, seguirá teniendo más pares que contienen la copia de información de todos en la red. Además, no está sujeto a los estándares idealistas de un sistema central, por lo tanto, no es propenso a la censura.

La estructura de red entre pares en las criptomonedas se rige de acuerdo con el mecanismo de consenso que es propuesto por la comunidad. Para criptomonedas como Bitcoin y Ethereum que utilizan un mecanismo de consenso de prueba de trabajo (Ethereum eventualmente pasará a Prueba de Apuesta), todos los nodos tienen el mismo privilegio, sin embargo, sus funciones y grado de participación pueden diferir. No hay un servidor o entidad centralizada, ni hay una jerarquía [1].

Estas criptomonedas descentralizadas están estructuradas de esa manera por una simple razón, para mantenerse fieles a su filosofía. La idea es tener un sistema monetario, donde todos sean tratados como iguales y no haya un organismo central, que pueda determinar el valor de la moneda en función de favorecer a un individuo o a un grupo de personas. Esto es cierto tanto para Bitcoin como para Ethereum [1].La red sigue el protocolo de descubrimiento de pares. Supongamos que Alice le envía 3 ETH (ether) a Bob. Los nodos más cercanos a ella se enterarán de esto, y procederán a decir a los nodos más cercanos a ellos, y luego les dirán a sus pares más cercanos y así sucesivamente [1], esta trasmisión de información es conocida como broadcast. En la Figura 2.1.2.1. Podemos ver un ejemplo de una red peer-to-peer

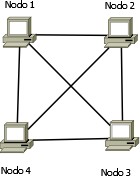
.

Figura 2.1.2.1: Red peer-to-peer

### 2.1.3 Bloque

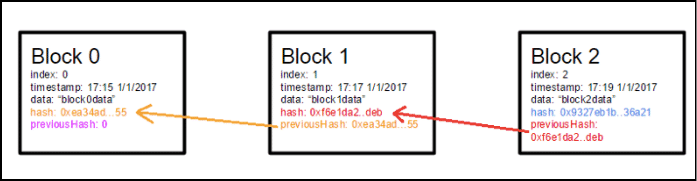
La blockchain se encuentra en crecimiento constante debido a que los usuarios que la componen agregan registros de datos que se definen como “bloques” o “eslabones” que están unidos entre sí, como lo muestra la Figura 2.1.3.2.

Figura 2.1.3.2: Composición de una cadena de bloque

Cuando se realizan transacciones en criptomonedas no se agregan directamente a la blockchain, sino que los mineros agrupan un puñado de transacciones y crean un candidato a bloque para ser agregado a la blockchain como un bloque valido. Cada bloque valido tiene un cabezal que funciona como los metadatos del mismo como lo muestra en la Figura 2.1.3.3.

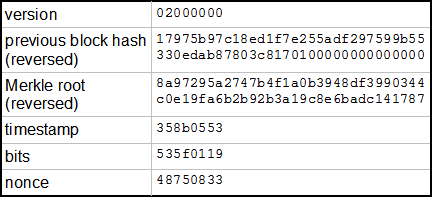


Figura 2.1.3.3 Cabezal de un bloque

En la Tabla 2.1.3.1 se muestra la definición de cada campo, el tamaño del campo y su formato de almacenamiento:

|  |  |  |  |
| --- | --- | --- | --- |
| **Campo** | **Descripción** | **Bytes** | **Formato de almacenamiento** |
| Versión | Describe la estructura de los datos dentro del bloque. Esto se usa para que las computadoras puedan leer el contenido de cada bloque correctamente. | 4 | Little-endian |
| Hash del bloque previo | Hash que indica el bloque anterior al que esta enlazado el bloque actual | 32 | Big-endian |
| Raíz de Merkle | Representa en un hash el  contenido de las transacciones presentes en el bloque | 32 | Big-endian |
| Timestamp | Indica la hora en que fue  creado el bloque | 4 | Little-endian |
| Bits | Es una versión más corta del hash target que los mineros encuentran para agregar el bloque | 4 | Little-endian |
| Nonce | Numero arbitrario que los mineros cambian para llegar al hash target y así poder agregar el bloque en la  blockchain | 4 | Little-endian |

Tabla 2.1.2.1: Campos del cabezal de un bloque.

En la blockchain existen distintos tipos de bloques que son los siguientes:

* + - * **Bloque Génesis:** Un bloque génesis viene a ser el primer bloque de una blockchain, de allí su nombre pues es el “génesis” (primero) de esa blockchain. La altura de este bloque por lo general es referida como el bloque cero.
      * **Bloques válidos:** Los bloques válidos son todos aquellos bloques que han sido minados e incluidos en la blockchain. Para ello, cada bloque minado debe pasar por el consenso de la red y reportarse como un bloque que ha logrado resolver el acertijo criptográfico, el cual consiste en encontrar un hash target que cumpla con la característica de tener una cierta cantidad de ceros a la izquierda por medio de funciones criptográficas. Una vez la red ha entrado en consenso el bloque es incluido en la blockchain y propagado por todos sus bloques [9].
      * **Bloques huérfanos:** Los bloques sueltos o huérfanos son bloques válidos que no forman parte de la blockchain. Estos suelen ocurrir cuando dos mineros producen bloques casi al mismo momento o pueden ser causados por atacantes con suficiente poder de cómputo para tratar de revertir una transacción. En este momento, el protocolo de consenso de la red entra en acción para decidir qué bloque será validado (incluido en la cadena) y cual será dado por huérfano, eventualmente la elección del bloque se da por la cadena más larga [9].

### 2.1.4 Funcionamiento

La blockchain es un método ingenioso para transmitir información de un punto A hasta un punto B de manera automatizada y segura. Cuando un miembro de la blockchain inicia un proceso se crea un eslabón o bloque. Este bloque es verificado por los demás miembros de la blockchain y luego es distribuido en todos los nodos. Cuando el bloque es verificado correctamente, es agregado en la blockchain y este es replicado en toda la cadena, creando un registro y un historial único.

Falsificar un bloque de la blockchain es improbable pero se puede dar el caso en que un usuario o un grupo de usuarios con muchos recursos se apoderen del 51% de la tasa de hash lo que le permite al atacante o atacantes intentar revertir las transacciones que han realizado mientras estaban en control, un ataque mayoritario exitoso también permitiría al usuario maliciosos evitar que algunas o todas las transacciones sean confirmadas (también conocida como denegación de servicio por transacción) o evitar que algunos o todos los otros mineros realicen actividades mineras, lo que lleva al llamado monopolio minero. Ethereum emplea la blockchain para hacer transacciones por lo transparente de la misma, pero también se pueden utilizar para otras soluciones que incluyan blockchain.

Los registros de datos de la blockchain están constantemente actualizándose en busca de bloques nuevos que se guardan en tiempo real, cada nodo contiene una copia de cada bloque de la cadena, los bloques son verificables por todos los miembros de la cadena, en el caso de que un bloque pierda información u ocurra un error los datos son más fácil de recuperar.

Las principales ventajas de la blockchain son:

* Es descentralizada.
* La propiedad de los datos es traspasados de un usuaria A - B por medio de firmas criptográficas lo que asegura autenticidad.
* La cadena es inmutable, nadie puede modificar los datos ya registrados.
* La blockchain es transparente, cualquiera puede verificar los datos si así lo desea.

### 2.1.5 Características

Las tres propiedades principales de la tecnología Blockchain que la han ayudado a obtener una gran aceptación son las siguientes:

* Descentralización
* Transparencia
* Inmutabilidad

#### 2.1.5.1 Descentralización

Como se ha mencionado anteriormente, la cadena de bloque no requiere de una entidad centralizada que supervise el paso de información entre pares, debido al uso de las redes peer-to-peer, cada nodo que pertenece a la red es responsable del mantenimiento de la misma, cada uno de los nodos contiene una copia de información de toda la cadena, la información es fácil de verificar por todos los miembros de la cadena sin necesidad de un tercero en el medio [1].

#### 2.1.5.2 Transparencia

Uno de los conceptos más interesantes e incomprendidos en la tecnología blockchain es la "transparencia". La identidad de una persona se oculta a través de una criptografía compleja y se representa solo por su dirección pública. Por lo tanto, si busca el historial de transacciones de una persona, no verá "Bob envió 1 BTC" sino que verá "1MF1bhsFLkBzzz9vpFYEmvwT2TbyCt7NZJ envió 1 BTC" [1].

En la Figura 2.1.5.4 se representa este ejemplo extraído de la blockchain de Ethereum [1].



Figura 2.1.5.4: cadena de bloque Ethereum

La identidad real de la persona está asegurada, sin embargo, aún verá todas las transacciones realizadas por su dirección pública. Este nivel de transparencia nunca antes había existido dentro de un sistema financiero. Hablando únicamente desde el punto de vista de las criptomonedas, si se obtiene la dirección pública de una de estas grandes empresas, puede ser ingresada en un explorador de bloque para visualizar todas las transacciones en las que se ha visto involucrada [1].

Debido a la naturaleza inmutable de la blockchain y a la seguridad de da su transparencia, ningún nodo puede falsificar o modificar los registros ya almacenados en un bloque. No obstante, si esto llegara a ocurrir, la cadena expulsaría a ese nodo fraudulento debido a lo rápido y fácil que es verificar la cadena completa a través de cualquier nodo confiable.

#### 2.1.5.3 Inmutabilidad

En el contexto de la blockchain nadie puede modificar la información una vez registrada. La blockchain gana esta propiedad por medio de una función hash. La función hash o hashing la cual se encarga de tomar una cadena alfanumérica de entradas de cualquier tamaño y aplicar una función de encriptación para generar una cadena alfanumérica diferente de salida. En la Figura **2.1.5.4** se muestra un ejemplo de función hash [2]:

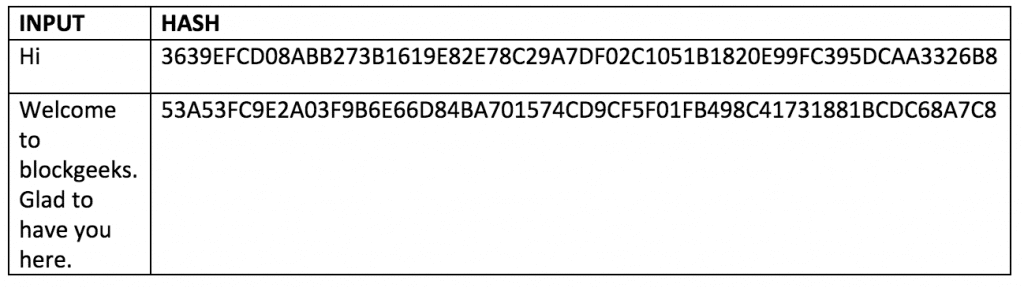


Figura 2.1.5.4: Ejemplo de función hash de SHA256.

La cadena de bloques es una lista vinculada que contiene datos y un puntero hash que está enlazado al bloque anterior, creando así la cadena. Si alguna transacción maliciosa cambiara el hash de una transacción registrada, al momento de la verificación por parte de los mineros el bloque generado será rechazado inmediatamente debido a que genera inconsistencia en la cadena, lo que hace improbable que la cadena sea alterada asegurando así la integridad de los datos.

### 2.1.6 Aplicaciones

* + - **Gobernanza:** Al hacer que los resultados sean totalmente transparentes y accesibles al público, la tecnología de bases de datos distribuidas podría aportartotal transparencia a elecciones o cualquier otro tipo de encuesta. Los contratos inteligentes basados en Ethereum ayudan a automatizar el proceso.La aplicación “Boardroom” permite la toma de decisiones organizacionales suceda en la cadena de bloques. En la práctica, esto significa que el gobierno de la empresa se vuelve totalmente transparente y verificable cuando se gestionan activos digitales, capital o información [1]**.**
    - **Cadena de suministros:** los consumidores siempre han querido que las afirmaciones éticas que hacen las compañías acerca de sus productos sean reales. Los libros distribuidos proporcionan una manera fácil de certificar que los artículos que compramos son genuinos. La transparencia viene con la marca de tiempo basada en blockchain que da fecha y ubicación, por ejemplo, que los productos perecederos no están caducados al momento del transporte, monitorear en todo momento el transporte del producto de la fábrica a los mercados.

La compañía conocida como Provenance con sede en el Reino Unido, ofrece a las auditorías de la cadena de suministro para una gama de bienes de consumo. Haciendo uso de la cadena de bloques Ethereum, un proyecto piloto de Provenance, garantiza que el pescado vendido en restaurantes de sushi en Japón haya sido cosecha de manera sostenible por sus proveedores en Indonesia [1].

### 2.1.7 Fork (Bifurcación)

Para que una cadena funcione adecuadamente y siga un estándar que todos los nodos deben cumplir se tienen que seguir unas reglas o protocolos para lograr un consenso entre todos.

Un fork ocurre cuando un miembro de la cadena quiere cambiar esas reglas en pos de mejorar la cadena o hacer una propia, existen 2 tipos de forks y estos son los siguientes:

* **Soft fork:** es una actualización del protocolo que permite compatibilidad con bloques no actualizados, ejemplo: cuando se va a cambiar el tamaño del bloque. El soft fork sucede cuando un nodo propone una actualización y los nodos que no son actualizados o no quieren actualizar, estos aún pueden seguir procesando transacciones sin problema.
* **Hard fork:** es una actualización que no permite compatibilidad. Todos los nodos se deben actualizar. Si esto no pasa, se creará una cadena nueva con los nodos que si actualizaron su protocolo generando una moneda diferente. Al crearse una cadena con una moneda diferente los bloques pertenecientes a la nueva cadena mantendrán las monedas de la cadena anterior.

### 2.1.8 Tipos de Blockchain

* + - **Blockchain pública:** una blockchain pública no tiene absolutamente ninguna restricción de acceso. Cualquier persona con conexión a internet puede actuar como un nodo participante. Este puede enviar transacciones y convertirse en un nodo validador. Para que dicha red siga operando y motivando a los mineros a seguir verificando bloques, la blockchain proporciona alguna forma de incentivo o recompensa económica. A menudo la cadena recompensa con la moneda nativa, pero a veces, pueden ser tarifas de transacciones verificadas [2].
* **Blockchain privada:** es una blockchain que requiere de permisos para poder interactuar con ella, ya que la participación de los nodos y el acceso del validador en esas blockchain está restringido. Cualquier usuario no puede unirse a una blockchain privada a menos que sea invitado por los administradores de la cadena. Este tipo de blockchain puede considerarse un punto medio para las empresas que están interesadas en la tecnología blockchain en general, pero no se sienten cómodos con el nivel de control ofrecido por redes públicas. Por lo general, las empresas buscan incorporar blockchain en sus procedimientos de contabilidad y mantenimiento de registros sin sacrificar la autonomía y correr el riesgo de exponer datos confidenciales en internet abiertamente [2].

## 2.2 Wallets (Billeteras digitales)

Es una interfaz que le permite al usuario interactuar con la blockchain a la que está conectada, le permite verificar saldo de monedas o recibir y transferir, la cartera interactúa con la cadena de bloque a través de direcciones que son guardadas en la misma, la dirección que guarda la cartera se denominan llave privada, con la llave privada se tiene acceso a la cartera y a los fondos almacenados en esa dirección.

A partir de la llave privada la cartera genera una dirección pública donde los usuarios pueden transferir monedas a cualquier cartera que soporte dicha moneda, algunas carteras cuando son creadas generan una frase semilla y a partir de esta derivan claves privadas con sus correspondientes claves públicas, si la cartera fuera a ser comprometida o destruida se puede introducir dicha frase para reconstruir la cartera.

### 2.2.1 Transferir Fondos

Para transferir fondos se requiere una combinación de cable pública y privada, para entender mejor este proceso tenemos el siguiente ejemplo: digamos que Usuario A quiere transferir unos bitcoins a Usuario B, seguidamente la cartera de Usuario A genera una dirección pública a partir de la llave privada y se la manda a Usuario B, entonces Usuario B va a la interfaz de usuario de la wallet donde le permite transferir, el Usuario B inserta la dirección pública dada y selección la opción de transferir, la cartera generara una solicitud de transferencia que contendrá información del emisor, del receptor, la cantidad de monedas transferidas y una firma del propietario que es generada a partir de procesos matemáticos con la llave privada del emisor; esto se hace para asegurar la propiedad de las monedas y pasarle la propiedad a la persona a quien se le transfiere. La solicitud es verificada y procesada por los mineros, si no hay nada malo con la transacción los fondos se bloquean para que solo la persona a la que se le transfiere pueda desbloquear y recibir los fondos. Usuario A usa su llave privada para generar una firma digital para desencriptar su dirección pública generada anteriormente y visualizar a los fondos [2].

### 2.2.2 Tipos

Lo más importante que tiene que proteger una wallet es la clave privadas del usuario, además debe ser usable en cualquier situación para hacer transacciones, con estos criterios de seguridad y conveniencia se desarrollaron distintos tipos de wallets según las necesidades del usuario.

Existen dos categorías de wallets, ellas son de almacenamiento de datos en frio y almacenamiento de datos en caliente.

### 2.2.3 Almacenamiento en frio

Son wallets que no tiene ningún acceso a internet como las siguientes:

**•Hardware Wallet:** Las billeteras de hardware son dispositivos físicos que almacenan claves privadas de forma segura. Por lo general, vienen en forma de una unidad flash que se puede conectar a una computadora para interactuar con ellas [7].

Las billeteras de hardware están diseñadas para proteger su clave privada incluso si se conecta con un dispositivo que está comprometido por malware. Incluso pueden ser usadas en una computadora pública cualquiera. Para que una persona envíe bitcoins a alguien que use una billetera de hardware, esta debe conectar la billetera de hardware a una computadora y usar algún tipo de software o página web que permita manipular la billetera. Las billeteras de hardware ofrecen la combinación óptima entre seguridad y facilidad de uso. Sus únicas limitaciones son que necesita tener la billetera de hardware para enviar las monedas y que además pueden llegar a ser costosas [7].

* + - * **Paper Wallets (Carteras de papel):** son solo trozos de papel con una clave privada o semilla escrita en ellos. Como mantiene su clave privada en una hoja de papel, solo alguien que tenga acceso a la clave privada contenida en el papel de respaldo podrá robar tus tokens. Sin embargo, las billeteras de papel se destruyen fácilmente y es recomendable crear varias copias para que, si se pierde una, tus tokens aún puedan recuperarse [7].

### 2.2.4 Almacenamiento en caliente

Son wallets que tienen acceso a internet como las siguientes:

* **Desktop Wallets (carteras de escritorio):** billeteras que almacenan claves privadas en una computadora. Mientras la computadora esté libre de malware o cualquier debilidad de seguridad, tus monedas o tokens estarán seguros. Sin embargo, todos sabemos que ese no es el caso para la mayoría de nosotros. Hoy es difícil estar 100% protegido y esto hace que las billeteras de escritorio sean un objetivo valioso para los piratas informáticos [7].

**•Web Wallets (carteras en línea):** los mercados, intercambios, sitios de apuestas y otros servicios de Bitcoin con frecuencia requieren que depositen fondos en sus billeteras en línea para llevar a cabo un negocio. Estas billeteras web son la opción menos segura para almacenar tokens, porque los operadores poseen la clave privada de los tokens almacenados en su sitio. En otras palabras, son carteras de custodia [7]. Las billeteras web también son más vulnerables a los piratas informáticos, por otro lado, también son muy convenientes ya que permiten comprar, vender y enviar bitcoins en cualquier momento [7].

Los servicios de billetera web más competentes proporcionarán opciones de autenticación de dos factores, como validar cada inicio de sesión de la cuenta con un mensaje de texto, para protegerse contra piratas informáticos externos. Aun así, para almacenar cualquier cantidad significativa de monedas, se debe estar consiente que si el sitio web es atacado todos tus tokens podrían ser robados, por eso es recomendable que se evite a toda costa guardar grandes cantidades de tokens en una billetera web solo guardando pequeños montos [7].

* **Mobile Wallet (cartera mobil):** estas billeteras almacenan su clave privada en un teléfono móvil y son las menos seguras de todas. Como los teléfonos se pierden, rompen o son robados con frecuencia, se recomienda habilitar la autenticación de dos factores: contraseña y copia de seguridad de la clave privada.

### 2.2.5 mecanismos de consenso

El hecho de que Blockchain sea un sistema distribuido requiere que sus nodos lleguen a un consenso mientras se ejecuta el sistema y se mantienen seguros sus datos. No obstante, deben respetarse una serie de protocolo para mantener la consistencia del sistema.

#### 2.2.5.1 Proof-of-work (PoW)

Proof-of-work (prueba de trabajo) se aplica cuando ocurre una nueva transacción en la blockchain. Para que un bloque nuevo sea agregado necesita ser verificado y para esto es necesario calcular un problema matemático. Los mineros compiten entre sí para resolver dicho problema, el primero que resuelva el problema recibirá una recompensa por el trabajo que aporto al agregar el bloque, el bloque agregado es fácil de verificar por los demás nodos de la cadena.

Este enfoque requiere una gran cantidad de energía eléctrica y tiempo para que un minero encuentre el hash objetivo para agregar el bloque a la cadena, esto garantiza la seguridad de la red ya que para realizar un ataque se requiera grandes cantidades de recursos generándole perdidas al atacante [3].

#### 2.2.5.2 Proof-of-stake (PoS)

Proof-of-stake (prueba tenencia) es una alternativa para llegar a un acuerdo [3].

Fue propuesto por un usuario del foro de Bitcointalk en 2012 [22] porque PoW requería demasiado tiempo, energía eléctrica y los mineros sintieron que extraer un solo bloque era un desperdicio de recursos [3].

Además, algunos estudios han sugerido que ejecutar y mantener redes PoW (como Bitcoin) es tan costoso como alimentar millones de hogares en los EE. UU. Alternativamente, PoS es una alternativa mucho más amigable con usuario y con el medio ambiente que PoW [3].

PoS usa un proceso pseudo-aleatorio para elegir al usuario que validará al bloque entrante basado en una combinación de criterios que pueden incluir el tamaño de la apuesta que tengan los bloques y el tiempo de antigüedad de una apuesta. En PoS se dice que los bloques son forjados, no minados, ya que este algoritmo es muy usado por monedas pre-minadas en vez de ser generadas minando.

Los usuarios que quieran participar en el proceso de forjar bloques tienen que bloquear cierta cantidad de las monedas que son parte de su apuesta, el tamaño de la apuesta aumenta la probabilidad de ser elegido para forjar el bloque entrante, mientras más grande la apuesta mayor es la probabilidad; cuando un bloque es elegido este verifica la transacción y lo agrega a la cadena , recibiendo como recompensa la comisión de esa transacción, si un nodo quiere dejar de forjar bloques su apuesta será desbloqueada con sus ganancias permitiéndole a la cadena verificar que no hay bloques fraudulentos.

Para no favorecer a los nodos más ricos el algoritmo tiene varios métodos de selección. Un método de selección que usa la cadena es seleccionar un nodo de manera aleatoria fijándose en la combinación del hash del bloque verificador y la cantidad de su apuesta, como las apuestas son públicas se puede predecir cual nodo será elegido.

Otro método es la selección por antigüedad de la apuesta, se selecciona la apuesta basándose en el número de días que la apuesta fue bloqueada y el número de monedas en la apuesta, cuando un nodo forja un bloque su vejez se reinicia y tendrá que esperar un tiempo a ser elegido, esto se usa para no favorecer a los nodos con apuestas grandes.

La apuesta funciona como un motivador financiero para el usuario, si este quiere aprobar transacciones fraudulentas la cadena lo detectará y se le descontarán monedas de su apuesta y hará que el nodo fraudulento pierda más dinero del ganado.

#### 2.2.5.3 Proof-of-Burn (PoB)

Proof-of-Burn (Prueba de derroche) es un algoritmo de consenso alternativo que trata de abordar el problema del alto consumo de energía eléctrica de un sistema PoW [4].

PoB a menudo se llama sistema PoW sin desperdicio de energía eléctrica. Funciona por el principio de permitir a los mineros "quemar" tokens. Luego se les otorga el derecho de escribir bloques en proporción a las monedas quemadas [4].

Iain Stewart, el inventor del algoritmo PoB, utiliza una analogía para describir el algoritmo: “las monedas quemadas son como plataformas mineras”. En esta analogía, un minero quema sus monedas para comprar una plataforma de minería virtual que les dá el poder de extraer bloques. Cuantas más monedas queme el minero, más grande será su "plataforma" minera virtual [4].

#### 2.2.5.4 Delegate-Proof-of-Stake (DPoS)

Delegated-Proof-of-Stake (Prueba de Participación Delegada) es un protocolo de consenso diseñado para blockchain altamente escalable. Este algoritmo fue creado por Daniel Larimer en el año 2014. La implementación de este protocolo ofrece Tolerancia a Fallas Bizantinas, esto significa que brinda altos niveles de seguridad para uso en blockchain públicas, además, su modelo de funcionamiento garantiza altos niveles de escalabilidad [5].

El algoritmo consiste en que los usuarios elegirán a un delegado mediante una votación, la cadena postula a los delegados dependiendo de la cantidad de la apuesta que posean, un delegado postulado puede dar parte de su apuesta a otro usuario para que vote a su favor, la cantidad de delegados que pueden elegir sería entre 27 y 120, ya que si hay muchos delegados la red esta sería muy lenta.

Ya con los delegados elegidos por la red, estos estarán programados en orden para que cada uno pueda verificar un bloque, el delegado que no lo verifique en el tiempo establecido será reprogramado en otro orden de verificación, si el delegado no verifica en el tiempo establecido muy seguido o se comporta de manera fraudulenta, se le descontará parte de su apuesta y nunca podrá ser elegido delegado de nuevo, por lo tanto, su reputación en la cadena bajará y será remplazado por otro delegado. En caso de que hallan muchos fallos en la cadena se elegirá la cadena más grande.

#### 2.2.5.5 Proof-of-Capacity (PoC)

Proof-of-capacity (prueba de capacidad) ofrece a los nodos de la red blockchain la posibilidad de utilizar el espacio vacío de sus discos duros para minar criptomonedas.

En lugar de modificar repetidamente los números del encabezado del bloque con el fin de encontrar el hash correcto, el PoC crea una lista de posibles soluciones en el disco duro del minero, incluso antes de iniciar el proceso de minería. Cuanto mayor sea la cantidad de memoria del disco duro, más soluciones posibles se pueden almacenar en él, lo que aumenta las posibilidades de que el minero tenga en su lista el valor hash deseado y por lo tanto obtenga una recompensa por el bloque.

Para hacer una analogía, si los premios de la lotería se basan en tener la mayoría de los números del décimo ganador, entonces un jugador con una lista más larga de posibles soluciones tendrá más probabilidades de ganar, además, el jugador puede guardar sus cartones de lotería y usarlos una y otra vez. La prueba de capacidad consta de dos etapas: un trazado (preparación del disco duro) y minería [5].

#### 2.2.5.6 Comparación

Cualquiera de los mecanismos de consenso mencionados anteriormente puede ser utilizado en distintas cadenas con distintas monedas, a medida que pasa el tiempo se seguirá desarrollado nuevos algoritmos de consenso con tal de mejorar la eficiencia de la cadena y el consumo de recursos.

Cada quien puede implementar el algoritmo que quiera en su cadena, con sus pros y contras como el PoW que fue el primer algoritmo de consenso desarrollado y el más común pero con una desventaja de consumo de energía eléctrica, el PoC trata de mejorar ese consumo de energía eléctrica pero no en gran medida; no obstante, también se pueden utilizar PoS o DPoS o PoB que son más amigables para los usuarios porque requieren participación activa en la cadena y mitiga en gran medida el consumo de energía eléctrica [2].

En tabla 2.2.5.6.2 [2] se puede ver las comparaciones de los distintos algoritmos según su usabilidad (el algoritmo más frecuente en ser implementado en una blockchain), sostenibilidad (cuantos recursos consume para los nodos a largo plazo) y seguridad (nivel de seguridad).

|  |  |  |  |
| --- | --- | --- | --- |
| **Mecanismos de consenso** | **Sostenibilidad** | **Usabilidad** | **Seguridad** |
| PoW | No sustentable | alta | alta |
| PoS | sustentable | baja | Media |
| DPoS | sustentable | baja | baja |
| PoB | Parcialmente sustentable | alta | alta |
| PoC | sustentable | alta | media |

Tabla 2.2.5.6.2: comparación de mecanismos de consenso.

## 2.3 Etherium

Ethereum es una plataforma de código abierto que posee una infraestructura computacional que ejecuta programas llamados contratos inteligentes. Utiliza una cadena de bloques para sincronizar y almacenar los cambios de estado de un sistema, junto con una criptomoneda llamada ether para medir y restringir los costos de recursos de ejecución [8].

La plataforma Ethereum permite a los desarrolladores construir potentes aplicaciones descentralizadas e incorporar funciones económicas proporcionado alta disponibilidad, audibilidad, transparencia y neutralidad, también reduce o elimina la censura [8].

### 2.3.1 Comparación con Bitcoin

Aunque Ethereum tenga todos los componentes de una blockchain, su propósito no es hacer un sistema de monetización descentralizado, sino que su valor de moneda yace en el ether que es utilizada por la plataforma de máquinas virtuales de Ethereum para correr aplicaciones.

A diferencia de Bitcoin, que tiene un lenguaje de script muy limitado, Ethereum está diseñado para ser un propósito general, una blockchain programable que ejecuta una máquina virtual capaz de ejecutar código de complejidad arbitraria e ilimitada. El lenguaje de script de Bitcoin está, intencionalmente, restringido para una simple evaluación verdadera / falsa para desarrollar una aplicación, el lenguaje de Ethereum es Turing completo, lo que significa que Ethereum puede funcionar directamente como una computadora de propósito general [7].

### 2.3.2 Ventajas y desventajas

Una de las ventajas de Ethereum al trabajar con un lenguaje Turing Complete, es que permite utilizar estructuras de codificación repetitivas (bucles) posibilitando la creación aplicaciones más complejas de una manera más simple y eficiente. Como desventaja, esto implica disponibilidad de un mecanismo como el “Gas” para evitar que el sistema colapse ya que siempre hay que tener “Gas” disponible para que el aplicativo funcione [9].

Se conoce como “Gas” a la forma que tiene Ethereum de calcular el coste de las transacciones y no es ni más ni menos que el gasto computacional de procesar una transacción o contrato inteligente en la red y las necesidades de su almacenamiento.

Este método surge para evitar que se produzcan bucles infinitos que traigan como consecuencia el colapso del sistema. De no existir el “Gas”, nada impediría que una aplicación mal programada pudiera dejar el sistema inservible [9].

### 2.3.3 Ethereum y aplicaciones Descentralizadas DApps

Ethereum comenzó como una forma de hacer una cadena de bloques de uso general que podría implementarse en una gran variedad de proyectos. La visión de Ethereum se expandió para convertirse en una plataforma para programación de DApps. Los DApps representan una perspectiva más amplia que los contratos inteligentes.

Un DApp es un contrato inteligente o una interfaz de usuario web. En términos más generales, un DApp es una aplicación web se basa en servicios de infraestructura abiertos, descentralizados y peer-to-peer.

Las Dapp se componen por lo general de:

• Smart contract (contratos inteligentes) [8].

• Una nueva interfaz de usuario web [8].

En algunas bibliografías notaremos la utilización del símbolo Ð en ÐApp es una letra del latín que significa ETH (ether), el término verdadero sería Ether aplicación [7]

### 2.3.4 Nodos Etherium

**•****Mist:** la Wallet oficial de Ethereum. Con ella se puede implementar contratos inteligentes, pero además sirve, entre otras cosas, para almacenar Ether, enviar/recibir transacciones e interactuar con la blockchain, ya sea en la red principal (mainnet) o en las redes de prueba (testnet) [9].

**•Geth:** es una herramienta de línea de comandos multipropósito que ejecuta un cliente Ethereum completo y está implementado en Go, una de las tres implementaciones originales del protocolo Ethereum junto con C++ y Python [9].

**•Parity:** es otro cliente Ethereum, parecido a Geth, aunque es más complejo su uso. Está creado por uno de los fundadores de Ethereum, Gavin Wood, y está escrito en el lenguaje de programación de bajo nivel Rust. Además, Parity es un entorno perfecto para la creación de DAplicaciones y contratos inteligentes gracias a su tecnología y su entorno destinado para desarrollo en Solidity, en el que se puede acceder a través de un navegador web [9].

**•Metamask:** es una extensión para los navegadores Google Chome, Opera, Firefox y Brave cuya función es la de ejecutar DAplicaciones basadas en Ethereum desde el navegador. Funciona como Wallet a la cual es posible transferir Ether e interactuar con aplicaciones descentralizadas como CriptoKitties. Además, es una herramienta que permitirá añadir smart contracts de Solidity [9].

### 2.3.5 Moneda de Etherium

La moneda de Ethereum se llama ether, identificado como “ETH” también se puede identificar con el símbolo Ξ o en menor medida ♦ (ej: 1 ETH, 1 Ξ, 1 ♦).

El ether se subdivide en unidades más pequeñas, la unidad más pequeña posible se llama wei. Un ether es 1 quintillón de wei (1 \* 10^18 o 1,000,000,000,000,000,000). Es posible que las personas se refieran a la moneda utilizando el término "Ethereum", pero este es un error común. Ethereum es el sistema y el ether es la moneda [8].

El valor del ether siempre se representa internamente en Ethereum como un valor entero sin signo conocido en wei. Cuando realiza una transacción con 1 ether, la transacción codifica 1,000,000,000,000,000,000 Wei como el valor. En la Tabla 2.3.5.3 muestra las distintas conversiones de wei y ether:

|  |  |  |  |
| --- | --- | --- | --- |
| **Valor en wei** | **Exponente** | **Nombre común** | **Nombre SI** |
| 1 | 1 | wei | Wei |
| 1000 | 10^3 | babbage | Kilowei o femtoether |
| 1000000 | 10^6 | lovelace | Megawei o picoether |
| 1000000000 | 10^9 | shannon | Gigawei o nanoether |
| 1000000000000 | 10^12 | szabo | Microether o micro |
| 1000000000000000 | 10^15 | finney | Milliether o milli |
| 1000000000000000000 | 10^18 | Ether | Ether |
| 1000000000000000000000 | 10^21 | Grand | Kiloether |
| 1000000000000000000000000 | 10^24 | Grand | Megaether |

Tabla 2.3.5.3: Conversión de la moneda de Ethereum

## 2.4 Contratos inteligentes (Smart Contracts)

Un smart contract es un tipo especial de instrucciones que es almacenada en la blockchain. Además, tiene la capacidad de auto ejecutar acciones de acuerdo con una serie de parámetros ya programados. Todo esto de forma inmutable, transparente y completamente segura.

Como parte de su nombre lo indica es un contrato pero que no está atado a terceros para confirmar su validez, en cambio, un contrato inteligente es capaz de ejecutarse y hacerse cumplir por sí mismo, de manera autónoma y automática, sin mediadores. Evitan el inconveniente de la interpretación al no ser verbal o escrito en los lenguajes que hablamos. Los smart contracts se tratan de “scripts” (códigos informáticos) escritos con lenguajes de programación. Esto quiere decir que los términos del contrato son puras sentencias y comandos en el código que lo forma.

Por otro lado, un smart contract puede ser creado y llamado por personas físicas y/o jurídicas, pero también, por máquinas u otros programas que funcionan de manera autónoma. Un smart contract tiene validez sin depender de autoridades, esto se debe a su naturaleza: es un código visible por todos y que no se puede cambiar al existir sobre la tecnología blockchain. Esto le confiere un carácter descentralizado, inmutable y transparente.

### 2.4.1 Lenguajes

Existen una gran variedad de lenguajes para desarrollar contratos inteligentes, en este proyecto nos concentraremos y profundizaremos en los leguajes más común para programar en Ethereum el cual es Solidity además vamos a definir otros leguajes que también son una propuesta para programar contratos inteligentes y estos son:

* + - **Serpent:** es un lenguaje de contrato inteligente basado en Python. Python es posiblemente el mejor lenguaje para programadores principiantes y el lenguaje más productivo para desarrolladores. Serpent se está utilizando actualmente para proyectos empresariales complejos. Fue desarrollado originalmente para Ethereum. También es compatible con el sistema Ethereum Classic (ETC) [12].
    - **Vyper:** es un lenguaje de contrato inteligente basado en Python 3 que implementa la máquina virtual de Ethereum. Vyper está avanzado en implementar PoS para la siguiente transición de Ethereum a ese mecanismo de consenso y proporcionar una rápida lectura y escritura de contratos inteligentes, con enfoque en la audibilidad, la simplicidad sintáctica y la sencillez.
    - **Solidity:** Solidity es un lenguaje de programación de alto nivel cuya síntesis es in hibrido entre C++ y JavaScript.

Los contratos se compilan en código de bytes (bytecode) para crear y desarrollar contratos inteligentes que se ejecuten en la Máquina Virtual Ethereum (EVM de sus siglas en inglés) [9].

Solidity es un lenguaje de programación de alto nivel significando que es un lenguaje “Turing Completo” [9].

Dicho de otro modo, el término Turing Completo aplicado a la tecnología Blockchain y especialmente a los contratos inteligentes, se refiere a la capacidad que tiene un lenguaje informático para resolver cualquier problema computacional y añadir reglas complejas como los bucles [9].

### 2.4.2 Solidity

Solidity es un lenguaje de alto nivel para la cadena de bloques pública de Ethereum y todas las aplicaciones basadas en Ethereum están escritas en este lenguaje. Solidity posee una sintaxis muy similar a JavaScript, pero con algunas diferencias.

Solidity utiliza una gran cantidad de conceptos de programación que existen en otros lenguajes. Por ejemplo, Solidity tiene variables, funciones, clases, operaciones aritméticas, manipulación de cadenas, etc. Mientras que, en un lenguaje como C, un programador probablemente crearía alguna forma de función "main", como "int main (arg1, arg2) {// code}"[23], Solidity funciona con un "contratc" que se crea de manera similar a C con “main” [23].

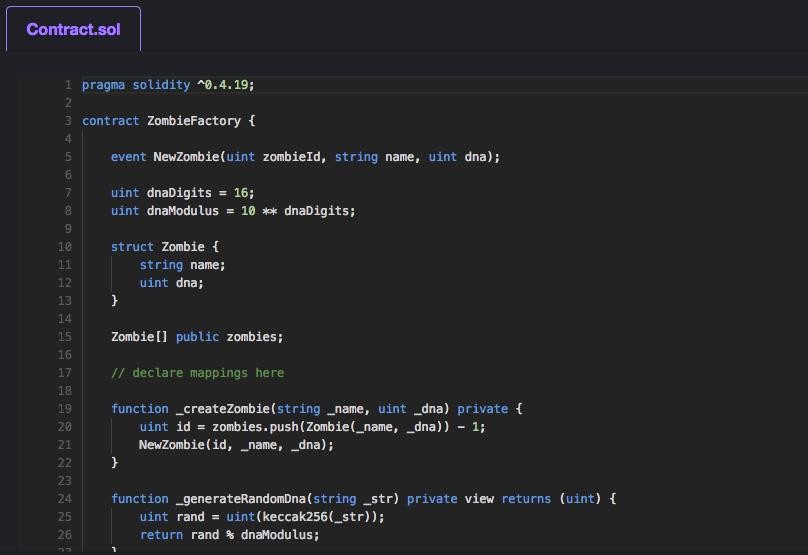


Figura 2.4.2.5: Ejemplo de código de solidity

Como se puede ver en la Figura 6.1.1.1 el código de Solidity se ve sorprendentemente similar a C ++, C # o JavaScript. Los desarrolladores de Python notarán que en Solidity, las variables necesitan que su tipo sea declarado explícitamente [23].

### 2.4.3 Herramientas de desarrollo

El mundo de las criptomonedas tuvo sus inicios con la creación de Bitcoin en 2009 y junto a él llegó la tecnología Blockchain: una base de datos distribuida que contenía los registros de todas las transacciones que se realizaban en la red.

Pocos años más tarde, gracias a Ethereum, llegó un proyecto mucho más potente y con mayores usos que el de ser un sistema de pago y una moneda digital descentralizada, como en el caso de Bitcoin y así surgieron los llamados smart contracts.

Entonces surgió el lenguaje de programación Solidity basado en javasript y C++ para el desarrollo de contratos inteligentes y con él una gran cantidad de herramientas que amplían el repertorio de soluciones que pueden abarcar los contratos inteligentes, no solo una solución para transferencia de tokens, también puede ser una solución para cadena de suministros, registro de indicadores de una empresa, blockchain de las cosas, entre muchas otras. En los siguientes subcapítulos se definirán las herramientas pertinentes para el desarrollo de la propuesta de proyecto.

### 2.4.4 Entornos de desarrollo

* + - * **Remix:** anteriormente conocido como Browser Solidity es el IDE oficial de Ethereum, proporciona un entorno de desarrollo integrado (IDE) que permite escribir contratos inteligentes basados en Solidity. Es una solución perfecta si lo que se busca es: crear contratos inteligentes y depurar su ejecución, tener acceso al estado y las propiedades de los smart contracts que ya han creado previamente. reducir errores de codificación y aplicar las mejores prácticas, analizando previamente el código. Se puede utilizar Remix ordenador o de manera online [9].
      * **Ethereum Studio:** es otro IDE que se caracteriza por tener especialización proporcionando un acceso completo a la red Ethereum, esto lo consigue a través de un shell de comandos [9].
      * **Atom:** es un editor de texto moderno, accesible y a la vez modificable. En otras palabras, es una herramienta editable, que se puede personalizar para hacer cualquier cosa, pero que también se puede utilizar de forma productiva sin necesidad de tocar un archivo de configuración. En la sección de la web Atom paquetes se pueden encontrar paquetes como Etheratom, un paquete hackeable para el IDE Atom que usa la biblioteca web3.js para interactuar con un nodo Ethereum [9].
      * **Plugin para Visual Studio:** está diseñado para permitir el desarrollo de contratos inteligentes de Solidity en este IDE de Microsoft. Para ello primero hay que adquirir Visual Studio y luego pasar a instalar la extensión. Visual Studio es un entorno rico de programación, integrado para la creación de aplicaciones para Windows, Android y iOS, así como aplicaciones web modernas y servicios en la nube [9].

### 2.4.5 Marcos de desarrollo (Frameworks)

Se pueden trabajar con distintos marcos de desarrollo en conjunción de cualquier IDE que sea compatible con Solidity, los siguientes son:

* + - * **Truffle:** Framework autodenominado “la navaja suiza del ejército” para Ethereum. Un marco de desarrollo muy completo y muy utilizado entre los programadores de contratos inteligentes de Ethereum. Tiene una gama de herramientas que aportan un alto nivel de abstracción al momento del desarrollo de DApp como por ejemplo ganache y drizzle [9].
      * **Embark:** Junto con Truffle, los marcos de desarrollo más usados y extendidos en la red Ethereum. Ambos frameworks se encargan de compilar, desplegar e interconectar con los contratos inteligentes [9].
      * **Dapp:** Es una herramienta de línea de comandos sencilla para el desarrollo de contratos inteligentes. Apoya estos casos de uso común: gestión de paquetes, construcción del código fuente, pruebas de unidad e implementación simple de contratos [9].
      * **Ganache:** es un constructor rápido de una cadena de bloques personal de Ethereum que puede usar para ejecutar pruebas, comandos e inspeccionar el estado de la cadena. Permite ver el resultado del registro de la cadena de bloques interna de Ganache, incluidas las respuestas y otra información vital de depuración, configura la minería avanzada con un solo clic, estableciendo tiempos de bloque que se adapten mejor a sus necesidades de desarrollo y monitoreo rápido de todas las transacciones [9].
* **Brownie:** es un marco de desarrollo y prueba basado en Python para contratos inteligentes dirigidos a la máquina virtual Ethereum. La consola de Brownie se puede utilizar con scripts Solidity para hacer testing en un ambiente local y obtener información útil al probar transacciones de test desarrollada con Python.

### 2.4.6 Librerias y clientes

* + - * **Web3.py**: es una colección de librerías que nos permite interactuar con los clientes Ethereum mencionados anteriormente, ya sea de forma local (teniendo el cliente en nuestro propio ordenador) o de forma remota (estando el cliente instalado en otro ordenador) usando los protocolos HTTP o IPC. Web3 nos permite compilar, desplegar e interactuar con nuestros propios contratos inteligentes.
      * **Zeppelin Solidity u OpenZeppelin:** OpenZeppelin es una biblioteca utilizada para escribir contratos inteligentes seguros sobre Ethereum. También se puede crear aplicaciones distribuidas (DAplicaciones), protocolos y organizaciones (DAOs) escritos en Solidity, además tiene integrado el entorno de trabajo de Truffle.

### 2.4.7 API de Infura

Uno de los problemas al momento de desplegar un contrato inteligente en una de las muchas cadenas de bloques de Ethereum ya sea de testing o una cadena pública, es que el desarrollador debe descargar la copia de la mayoría de los bloques de la cadena en su dispositivo, digamos que un usuario desarrolla una DApp y no tiene un equipo muy potente para descargarse un nodo completo de Ethereum, lo que desmotivaría al desarrollador de entrar al mundo de las DApp, pero existe una solución desarrollada por ConsenSys llamada Infura, esta API permite al desarrollador abstraerse de armar una infraestructura blockchain para que se concentre en el desarrollo total de DApp sin preocuparse por un gran consumo de recursos.

La suite Infura API proporciona acceso instantáneo a través de HTTPS y WebSockets a las redes Ethereum e IPFS. Facilitando el desarrolla de en el marco de trabajo web3.py. También es un clúster de nodos de Ethereum alojado que permite a sus usuarios ejecutar su aplicación sin necesidad de que configuren su propio nodo o billetera de Ethereum. Por razones de seguridad, Infura no administra sus claves privadas, lo que significa que Infura no puede firmar transacciones en su nombre por lo que se usa en conjunto con Trufflle y clientes Ethereum como MetaMask para firmar los contratos.

La parte más conocida de la infraestructura de Infura es la red de clientes Ethereum que abarca cuatro redes de Ethereum: Mainnet, Ropsten, Rinkeby y Kovan. Infura permite conectarse a estos clientes Ethereum a través de una conexión HTTPS por medio de websockets. También posee un módulo para nodos Ethereum de IPFS y gateways IPFS, en la Figura 2.4.6.6 se puede apreciar cómo esta enlazada Infura a la infraestructura Ethereum.

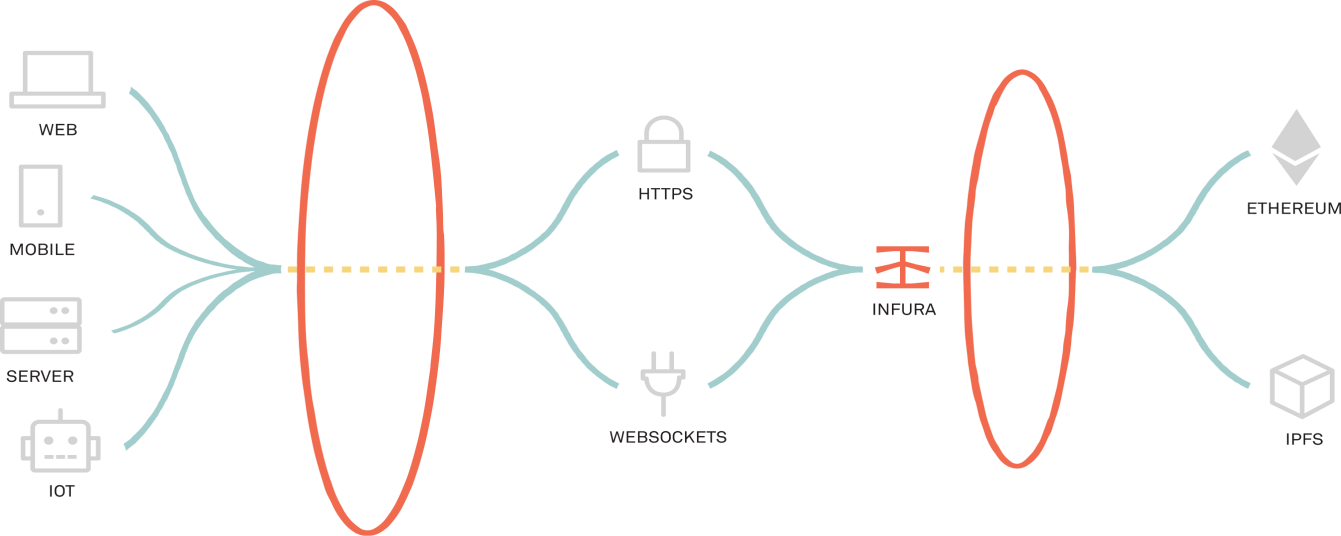


Figura 2.4.7.6: Enlace de Infura [12].

Algunas características más de Infura son:

* + - * Admite mainnet y testnets a través de JSON-RPC compatible con los clientes Ethereum, a través de HTTPS y WebSocket [12].
      * Soporta librerías estándar de IPFS que permiten almacenar y retirar data [12].
      * Permite conectar las DApp con el sencillo paso de agregar una simple URL predefinida de las redes Ethereum ya nombradas [12].
      * Se actualiza constantemente para una rápida comunicación entre los clientes Ethereum y mejora la escalabilidad de la red [12].
      * Se puede configurar dashboards para el monitoreo de la red [12].
      * Tiene un soporte 24/7 por la comunidad [12].

## Almacenamiento de datos

Como se dijo anteriormente la blockchain es un conjunto de registros almacenados en nodos que están interconectados entre sí, cada nodo puede guardar cierta cantidad de información, si alguna DApp genera una gran cantidad de transacciones por segundo entonces es posible que al momento de procesar tantas transacciones la cadena pueda saturarse, esto puede deberse al tamaño del bloque lo cual está regido por un protocolo que dicta su capacidad de almacenamiento de datos, pudiendo resultar en un ambiente perjudicial de trabajo donde el aplicativo no cumpla los estándares de desarrollo debido a la escalabilidad deficiente de la blockchain. Otro factor es el tiempo en que un nodo puede procesar un bloque, si partimos con que la verificación de un bloque es realizada por todos los nodos mineros en la cadena al mismo tiempo, esto limita la cantidad de datos que se pueden procesar por bloque, si ponemos de ejemplo a un usuario que quiere guardar una foto de 2MB y la cantidad promedio que puede procesar una blockchain Ethereum por bloque es de 20KB entonces se necesitaría al menos 100 bloques para procesar la imagen completa y si es que se reservan tantos bloques para una sola persona. Para evitar estos problemas de escalabilidad definiremos 2 soluciones de almacenamiento de datos que son IPFS (Sistema de archivos interplanetario) y swarm.

### 2.5.1 IPFS (Sistema de archivos interplanetario)

En la historia de internet la infraestructura que ha dominado en la de cliente- servidor. Los datos se almacenan en servidores centralizados y se accede a ellos mediante el direccionamiento basado en la ubicación. Esto hace que sea más fácil distribuir, administrar, proteger los datos y escalar la capacidad de los servidores y los clientes. Sin embargo, hay muchas debilidades en los ámbitos de seguridad, privacidad y eficiencia: el control del servidor se traduce en el control de los datos. Esto significa que cualquier persona con control del servidor puede acceder, alterar y eliminar sus datos; esto podría ser una entidad con autoridad legítima sobre el servidor o un hacker malicioso. En el direccionamiento basado en la ubicación, los datos se identifican por el lugar donde se ubican y no por su contenido. Esta limitación significa que debe ir a una ubicación específica para acceder a un dato, incluso si esos mismos datos están disponibles en algún lugar más cercano. Tampoco hay forma de saber si los datos han sido alterados, ya que el cliente solo necesita saber dónde está, no que son los datos.

IPFS llega como una alternativa a HTTP, IPFS sustituye la infraestructura cliente-servidor por una peer-to-peer de archivos compartidos. Los datos son guardados en los múltiples nodos de la red IPFS, posteriormente son identificados por una dirección fija que nunca cambia, esta ubicación sería en que nodo y en que carpeta está guardado el archivo, como si se tratara de un área local de archivos compartidos.

Para poder crear una red distribuida permanente IPFS se vale de un sistema de localización de contenido en vez de localización por direcciones como HTTP. Como, por ejemplo:

Así es como se ve una solicitud HTTP <http://10.20.30.40/folder/file.txt> Así es como se ve una solicitud IPFS /ipfs/QmT5NvUtoM5n/folder/file.txt

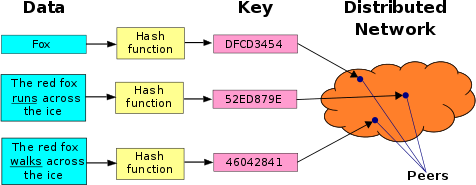
En lugar de ubicar la dirección del archivo, IPFS usa una representación del contenido en sí para direccionarlo, esto se hace usando un hash criptográfico en un archivo que sirve como dirección. El hash representa un objeto raíz y se pueden encontrar otros objetos en su ruta. En lugar de hablar con un servidor y obtener acceso a este para recibir los datos, es preferible conectarse directamente con la fuente. De esta manera, el sistema aprovecha la proximidad física. Si alguien muy cercano a mí tiene lo que quiero, lo obtendré directamente de él en lugar de conectarme a un servidor central y viceversa. IPFS logra esto por medio de las redes par-a-par, en la Figura 2.5.1.7 podemos ver un ejemplo de cómo funciona IPFS [13].

Figura 2.5.1.7: Funcionamiento de IPFS [13]

Siguiendo la lógica de IPFS y como se combinan esta con Ethereum se obtendrá un aumento enorme en escalabilidad. Recordemos que la blockchain permite registrar los datos y estos son inmutables, pero con la limitante de una baja capacidad de almacenamiento de datos, entonces IPFS llega para resolver este problema.

Si tomamos como ejemplo una foto digital y la guardamos en un nodo de la red IPFS, la red genera una dirección por medio del hash del archivo; como el hash de la dirección IPFS es de 46 bytes en tamaño, entonces en la blockchain Ethereum se guardaría el hash de dirección IPFS que lógicamente representa la foto que está guarda en alguna parte de la red, como el hash IPFS es único, al momento de agregarlo a Ethereum le otorga otra capa de seguridad, dándole fiabilidad a los datos a los que apunta la dirección IPFS.

#### 2.5.1.1 Componentes

* + - * **Content Identifiers (CIDs**)**:** Un identificador de contenido o CID, es una etiqueta utilizada para señalar objetos en IPFS. No indica dónde se almacena el contenido, pero forma una especie de dirección basada en el contenido mismo. Los CID son cortos, independientemente del tamaño de su contenido subyacente. Los CID se basan en el hash criptográfico del contenido. Eso significa: Cualquier diferencia en el contenido debería producir un CID diferente y el mismo contenido agregado a dos nodos IPFS diferentes usando la misma configuración deberían producir exactamente el mismo CID [14].
      * **Distributed Hash Tables (DHT):** las tablas de hash distribuidas (DHT) son almacenes de llaves distribuidos donde las llaves son hashes criptográficos. Los DHT están distribuidos en cada par de nodos y cada nodo es responsable de un subconjunto del DHT. Cuando un par recibe una solicitud, la responde o se pasa a otro hasta que se encuentre un par que pueda responderla. Dependiendo de la implementación, una solicitud no respondida por el primer nodo contactado puede ser reenviada de par-a-par, con el último par contactando al par solicitante o reenviada de par-a-par, con la respuesta reenviada siguiendo el mismo camino o respondida con la información de contacto de un nodo que tiene mejores posibilidades de poder responder [14].
      * **DNSLink:** DNSLink utiliza registros TXT de DNS para asignar un nombre de dominio (como ipfs.io) a una dirección IPFS. Debido a que puede editar sus registros DNS, puede usarlos para apuntar siempre a la última versión de un objeto en IPFS (recordando que la dirección de un objeto IPFS cambia si modifica si se modifica el objeto) [14].
      * **Hashes:** Los hashes son funciones que toman una entrada arbitraria y devuelven un valor de longitud fija. El valor particular depende del algoritmo hash que se esté usando, como SHA-1, SHA-256 o BLAKE2, pero un algoritmo hash siempre devuelve el mismo valor para una entrada dada [14].

**IPNS:** Sistema de Nombres Interplanetarios (IPNS) es un sistema para crear y actualizar enlaces mutables al contenido de IPFS. Dado que los objetos en IPFS tienen dirección de contenido, su dirección cambia cada vez que cambia su contenido. Eso es útil para una variedad de cosas, pero hace que sea difícil obtener la última versión de algo. Un nombre en IPNS es el hash de una clave pública. Está asociado con un registro que contiene información sobre el hash al que se vincula, que está firmado por la clave privada correspondiente [14].

* + - * **IPLD:** IPLD (Data model for Interoperable Protocols) ofrece una descripción más clara de cómo vincular diferentes tipos de estructuras basadas en hash (por ejemplo, vincular un archivo en IPFS a un commit en Git), tiene un formato más generalizado y flexible, y utiliza una representación compatible con JSON, entre otras mejoras [14].
      * **Pinning:** Los nodos IPFS tratan los datos que almacenan como una caché, lo que significa que no hay garantía de que los datos continúen almacenados. Anclar un CID le dice a un servidor IPFS que los datos son importantes y que no deben desecharse. Se debe anclar cualquier contenido que sea considerado importante, para asegurarse de que el contenido se conserve a largo plazo. Dado que los datos que una persona considere importantes para una persona pueden no ser importantes para otra, la fijación le permite tener control sobre el espacio en disco y la retención de datos que necesita [14].
      * **UnixFS:** Un archivo en IPFS no es solo contenido, puede ser demasiado grande para caber en un solo bloque, por lo que necesita metadatos para vincular todos sus bloques. Puede ser un enlace simbólico o un directorio, por lo que necesita metadatos para vincular a otros archivos. UnixFS es el formato de datos utilizado para representar archivos y todos sus enlaces y metadatos en IPFS, y se basa libremente en cómo funcionan los archivos en Unix.

### 2.5.2 Swarm

Swarm es una plataforma de almacenamiento de datos distribuido y un servicio de distribución de contenido; un servicio de capa base nativo de la pila Ethereum Web3. El objetivo es una solución de almacenamiento de datos y servicio de peer-to-peer que tenga cero tiempo de inactividad, sea resistente a ataques de denegación de servicios, tolerante a fallas y a censura, además de autosustentable debido a un sistema de incentivos incorporado. La capa de incentivos utiliza la cantidad de recursos aportada a la red peer-to-peer para el ancho de banda y la capacidad de almacenamiento de datos, a cada nodo se le dará una recompensa equivalente al aporte que ofrezca a la red. Swarm está diseñado para integrarse profundamente con la capa de red multiprotocolo devp2p (red peer-to-peer que implementa Ethereum) de Ethereum, así como con la cadena de bloques Ethereum para la resolución de nombres de dominio, pagos de servicios y seguro de disponibilidad de contenido. Los nodos en el testnet actual usan la cadena de prueba Ropsten solo para la resolución de nombres de dominio. El objetivo principal de Swarm es proporcionar almacenamiento de datos descentralizado y redundante del registro público de Ethereum, en particular el almacenamiento de datos y la distribución de datos y código Dapp, así como datos de blockchain [15].

Hay dos características principales que diferencian a Swarm de otras soluciones descentralizadas de almacenamiento de datos distribuido. Si bien los servicios existentes (Bittorrent, Zeronet, IPFS) le permiten registrar y compartir el contenido que aloja en su servidor, Swarm proporciona el alojamiento en sí como un servicio descentralizado de almacenamiento de datos en la nube. Hay una sensación genuina de que puede simplemente "subir y desaparecer": se sube el contenido al enjambre y es recuperado más tarde, todo potencialmente sin un disco duro. Swarm aspira a ser un servicio genérico de almacenamiento de datos y entrega, atiende a casos de uso que van desde servir aplicaciones web interactivas en tiempo real de baja latencia hasta actuar como almacenamiento de datos persistente garantizado para contenido poco utilizado [15].

Swarm opera con nodos que ejecutan un protocolo cableado llamado bzz (zumbido) que utiliza la pila de red Ethereum devp2p como transporte. El protocolo Swarm (bzz) define un modo de interacción. En esencia, Swarm implementa un almacén de fragmentos distribuidos llamados “chunks”. Los fragmentos son bloques de datos arbitrarios con un tamaño máximo fijo que es actualmente 4KB. El direccionamiento del contenido significa que la dirección de cualquier fragmento se deriva de la meta data de los datos. El esquema de direccionamiento recurre a una función hash que toma un fragmento como entrada y devuelve una clave larga de 32 bytes como salida. [15].

Este hash de un chunk es la dirección que los clientes pueden usar para recuperar el fragmento (la preimagen del hash). El direccionamiento irreversible y sin colisiones brinda protección de integridad de inmediato: el cliente puede saber si el fragmento está dañado o ha sido manipulado simplemente con el hash [15].

Swarm es un almacén de chunks distribuido es que puede cargar contenido en él. Todos los nodos que constituyen el enjambre dedican recursos (espacio en disco, memoria, ancho de banda y CPU) para almacenar y servir fragmentos. Los nodos de enjambre tienen una dirección (el hash de la dirección de su cuenta bzz) en el mismo espacio de llaves que los propios fragmentos. Llamemos a este espacio de direcciones la red superpuesta. Si cargamos un fragmento en el enjambre, el protocolo determina que eventualmente terminará siendo almacenado en los nodos más cercanos a la dirección del fragmento (de acuerdo con una medida de distancia bien definida en el espacio de direcciones superpuestas). El proceso por el cual los fragmentos llegan a su dirección se llama sincronización y es parte del protocolo. Los nodos que luego desean recuperar el contenido pueden encontrarlo nuevamente al reenviar una consulta a los nodos que están cerca de la dirección del contenido. De hecho, cuando un nodo necesita un fragmento, simplemente publican una solicitud en Swarm con la dirección del contenido y Swarm reenviará las solicitudes hasta que se encuentren los datos (o la solicitud agota el tiempo de espera) En este sentido, Swarm es similar a una tabla hash distribuida tradicional (DHT) [15].

#### 2.5.2.1 Componentes

* + - * **Chunks:** un chunk está representado por cualquier tipo de dato sea video, fotos, txt, etc. Cada chunk se conecta con un par de nodos en la red que luego son comprimidos en uno solo para generar un fragmento intermedio. Actualmente 128 hashes componen un nuevo chunk, como resultado, los datos están representados por un árbol de merkle, y el hash raíz del árbol actúa como la dirección que utiliza para recuperar el archivo cargado [15].
      * **Manifiesto:** Un manifiesto es una matriz json que tiene como entrada otros manifiestos. Una entrada especifica mínimamente una ruta, un tipo de contenido y un hash que apunta al contenido real. Los manifiestos permiten crear un sitio virtual alojado en Swarm, que proporciona direccionamiento basado en url siempre asumiendo que la parte del host de la url apunta a un manifiesto, y la ruta coincide con las rutas de las entradas del manifiesto. Las entradas de manifiesto pueden apuntar a otros manifiestos, por lo que pueden incrustarse recursivamente, lo que permite que los manifiestos se codifiquen como un archivo compactado que se escala de manera eficiente a grandes conjuntos de datos. Los manifiestos también pueden considerarse mapas de sitio o tablas de enrutamiento que asignan cadenas de URL al contenido. Dado que en cada paso del camino, hemos fusionado estructuras o direcciones de contenido, los manifiestos brindan protección de integridad para todo un sitio [15].
      * **Ethereum Name Service:** El Ethereum Name Service (ENS) permite a los usuarios registrar nombres legibles por humanos para su contenido. ENS se implementa como un contrato inteligente en la red Ethereum y puede considerarse el equivalente del servicio de nombres de dominio (DNS) que facilita la asignación de nombres de contenido en los servicios tradicionales de Internet [15].

## Placas de desarrollo

Una placa de desarrollo es un circuito impreso que contiene un microprocesador o microcontrolador, en el cual se posee la mínima lógica de soporte. Esto es para que los desarrolladores e ingenieros tengan la capacidad para familiarizarse con la placa y cómo se programa. Su propósito principal es ser usado para desarrollo de proyectos. Están hechas para ser la base o el cimiento del proyecto. Tal que es posible elegir a la que se ajuste más a las necesidades del usuario para la aplicación requerida.

Hay muchos tipos distintos de placas de desarrollo, algunas más orientadas para el IoT (Internet of Things), otros para Inteligencia artificial, poder de procesamiento, etc.

En este proyecto se utilizará la placa Heltec ESP32 y los sensores DHT-11 de temperatura y el módulo GPS Adafruit.

### 2.6.1 Heltec-ESP32

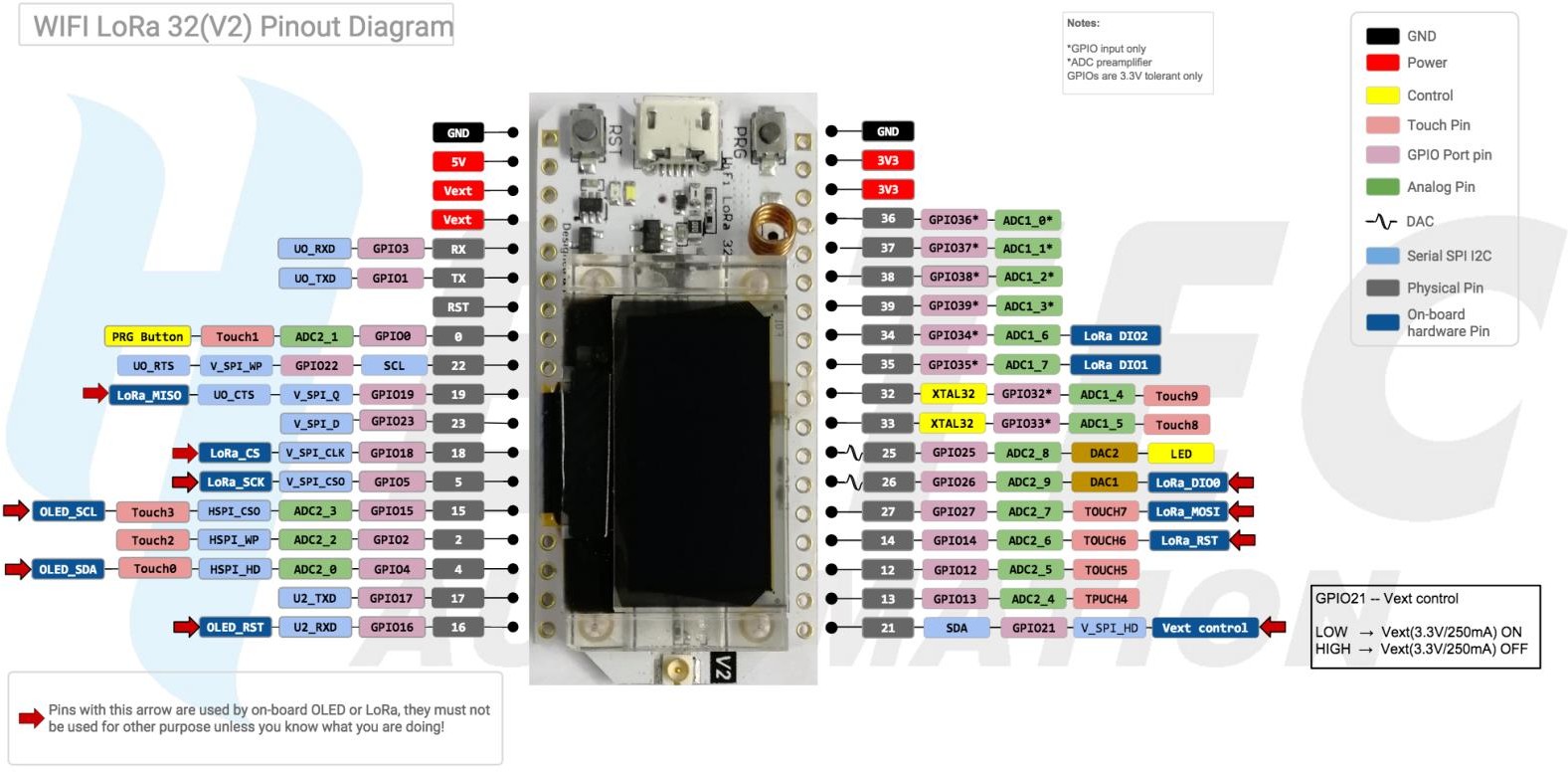
Este nodo es la integración de un ESP32 y un Chip LoRa SX1276 (868-915Mhz) Pantalla Oled de 0.96 pulgadas, convertidor usb-serial CP2102. En la Figura 2.6.1.8 se pueden apreciar cómo está compuesto el dispositivo [16].

Figura 2.6.1.8: dispositivo Heltec-ESP32[16]

Las especificaciones del dispositivo son [16]:

* + - Frecuencia máxima: 240 MHz Flash: 32M-Bits.
    - Procesador: Para Tensilica LX6 Dual Core.
    - Pantalla: Oled 0.96″ azul.
    - Maestro chip: ESP32.
    - LoRa chip: SX1276.
    - Banda de frecuencia compatible: 868-915 MHz.
    - Distancia de comunicación abierta: 2,8 Km.
    - Capacidad de procesamiento: hasta 600 MIPS.
    - Bluetooth de modo Dual: Bluetooth tradicional y Bluetooth BLE de baja potencia.
    - Entorno de desarrollo: soporte perfecto para Arduino, ESP-IDF y Mongoose OS.
    - Voltaje de funcionamiento: 3,3-7 V.
    - Rango de temperatura: -40-90°.
    - Sensibilidad del receptor: -139dBm (SF12, 125 KHZ).
    - Rendimiento continúo UDP: 135 Mbps.
    - Adaptador USB chip: CP2102.
    - Modo de soporte: rastreador, estación, softAP y Wi-Fi directo.
    - Potencia de transmisión: 19.5dBm @ 12b, 16.5dBm @ 12g, 15,5dBm @ 12n.
    - Velocidad de datos: 150 Mbps @ 12n HT40, 72 Mbps @ 12n HT20, 54 Mbps @ 12g, 12 Mbps @ 12b.

### 2.6.2 DHT-11

El DHT11 es un sensor digital de temperatura y humedad relativa de bajo costo y fácil uso. Integra un sensor capacitivo de humedad y un termistor para medir el aire circundante, y muestra los datos mediante una señal digital en el pin de datos (no posee salida analógica). Utilizado en aplicaciones académicas relacionadas al control automático de temperatura, aire acondicionado, monitoreo ambiental en agricultura y más [17].

No se debe confundir entre analógico y digital. Aunque lo conectemos a un pin digital, se trata de un dispositivo analógico. Dentro del propio dispositivo se hace la conversión entre analógico y digital.

Por lo tanto, se parte de una señal analógica que luego es convertida en formato digital y se enviará al microcontrolador. La trama de datos es de 40 bits correspondiente a la información de humedad, temperatura y redundancia de datos del DHT11, la Figura 2.6.2.9 muestra la trama de datos [17].



Figura 2.6.2.9: Trama de datos del DHT-11

El primer grupo de 8-bit es la parte entera de la humedad y el segundo grupo la parte decimal. Lo mismo ocurre con el tercer y cuarto grupo, la parte entera de la temperatura y la parte decimal. Por último, los bits de paridad para confirmar que no hay datos corruptos.

Lo único que hacen estos bits de paridad es asegurar que la información sea correcta, sumando los 4 primeros grupos de 8-bit. Esta suma debe ser igual a los bits de paridad. Si nos centramos en la Figura 8.2.1 y sumamos los bits, comprobamos que todo está correcto.

**0011 0101 + 0000 0000 + 0001 1000 + 0000 0000 = 0100 1001**

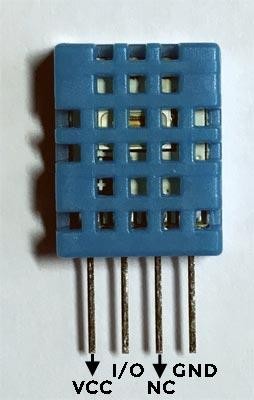
El dispositivo viene con 4 pines que están ilustrados en la Figura 2.6.2.10 y estos son [17]:

Figura 2.6.2.10: Dispositivo DHT-11[17]

* + - VCC: alimentación.
    - I/O: transmisión de datos.
    - NC: no conecta, pin al aire.
    - GND: conexión a tierra.

Las especificaciones técnicas son [17]:

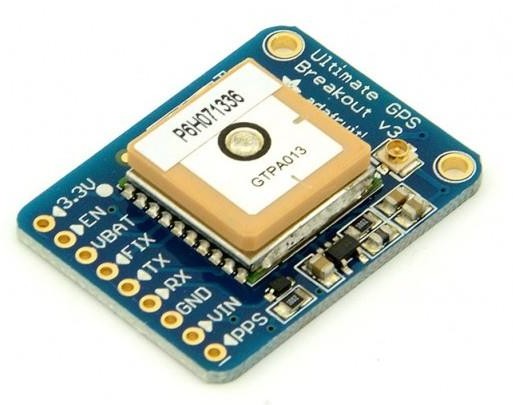
* + - * Voltaje de Operación: 3V - 5V DC.
      * Rango de medición de temperatura: 0 a 50 °C.
      * Precisión de medición de temperatura: ±2.0 °C.
      * Resolución Temperatura: 0.1°C.
      * Rango de medición de humedad: 20% a 90% RH.
      * Precisión de medición de humedad: 4% RH.
      * Resolución Humedad: 1% RH.
      * Tiempo de censado: 1 seg.
      * Interface digital: Single-bus (bidireccional).
      * Modelo: DHT11.
      * Dimensiones: 16\*12\*5 mm.
      * Peso: 1 gr.
      * Carcasa de plástico celeste

### 2.6.3 Adafruit GPS

Módulo que permite obtener coordenadas de posicionamiento mediante servicio satelital GPS. Está basado en el MTK3339, que permite seguir hasta 22 satélites por 66 canales, cuanta con una gran sensibilidad (-165dB) y permite realizar 10 actualizaciones de posición por segundo [18].

Al contar con un regulador de 3,3 V se puede alimentar a 3,3V y a 5V DC, además incorpora un RTC (Real Time Clock) para incorporar la hora exacta a las localizaciones. Además de contar con una antena interna dispone de un conector UFL para acoplar una externa y es el propio modulo el que elige la antena adecuada [18].

Finalmente permite hacer data-logging, la memoria flash del módulo permite almacenar hasta 16 horas de datos de localización [18]. En la Figura 2.6.3.11 podemos visualizar el dispositivo.

Figura 2.6.3.11: Dispositivo Adafruit GPS[18]

Las especificaciones técnicas son [18]:

* + - Satélites: de 22 a 66 búsquedas.
    - Tamaño de antena de parche: 15 mm x 15 mm x 4 mm.
    - Velocidad de actualización: 1 a 10 Hz.
    - Precisión de posición: < 3 metros (toda la tecnología GPS tiene una precisión de aproximadamente 3 m).
    - Precisión de velocidad: 0.1 metros / s.
    - Inicio cálido / frío: 34 segundos.
    - Sensibilidad de adquisición: -145 dBm.
    - Sensibilidad de seguimiento: -165 dBm.
    - Velocidad máxima: 515 m / s.
    - Rango de Vin: 3,0 - 5,5VDC.
    - MTK3339 Corriente de funciousuario Aento: seguimiento de 25 mA, consumo de corriente de 20 mA durante la navegación.
    - Salida: NMEA 0183, 9600 baudios por defecto, salida lógica de nivel de 3V, entrada segura de 5V.
    - DGPS / WAAS / EGNOS compatible.
    - Cumplimiento de FCC E912 y soporte AGPS (Modo sin conexión: EPO válido hasta 14 días).
    - Hasta 210 canales PRN.
    - Detección y reducción de ruido.
    - Detección de múltiples rutas y compensación.

## Redes IoT

La definición de IoT son la agrupación e interconexión de dispositivos y objetos a través de una red (bien sea privada o Internet, la red de redes), dónde todos ellos puedan interactuar entre sí. Respecto al tipo de objetos o dispositivos podrían ser cualquiera, desde sensores y dispositivos mecánicos hasta objetos cotidianos como una nevera o tostador, etc. Cualquier cosa que pueda ser conectada al internet e interactuar con el entorno sin necesidad de la intervención humana, el objetivo por tanto es una la interacción de máquina a máquina, o lo que se conoce como una interacción M2M (machine to machine) o dispositivos M2M.

Una Red del Internet de las Cosas (NB-IoT) es una Red de Baja Potencia de Banda Ancha (LPWAN), es un estándar de tecnología de radio desarrollado por 3GPP para habilitar servicios de alta gama para dispositivos móviles.

NB-IoT se enfoca específicamente en la cobertura de espacios cerrados, de bajo costo, batería de larga duración, y densidades de conexión altas. NB-IoT utiliza una ampliación de la red LTE estándar, pero limita el ancho de banda a una sola banda de 200kHz, también, utiliza, la modulación OFDM (Multiplexación por división de frecuencias ortogonales) para comunicaciones de pocas conexiones entre dispositivos y utiliza SC-FDMA (Acceso múltiple por división de frecuencia de portadora única) para comunicaciones con más conexiones entre dispositivos.

### 2.7.1 LoRaWAN

Es una especificación para redes de baja potencia y área amplia, LPWAN (Low Power Wide Area Network), diseñada específicamente para dispositivos de bajo consumo de alimentación, que operan en redes de alcance local, regional, nacionales o globales.

El estándar de red LoRaWAN apunta a requerimientos característicos de Internet de las Cosas, tales como: conexiones bidireccionales seguras, bajo consumo de energía, largo alcance de comunicación, bajas velocidades de datos, baja frecuencia de transmisión, movilidad y servicios de localización. Permite la interconexión entre objetos inteligentes sin la necesidad de instalaciones locales complejas, y además otorga amplia libertad de uso al usuario final, al desarrollador y a las empresas que quieran instalar su propia red IoT.

Las comunicaciones entre los dispositivos y el servidor de red, son generalmente unidireccionales o bidireccionales, pero el estándar también soporta multidifusión, permitiendo la actualización de software de forma inalámbrica.

La comunicación entre dispositivos finales y las puertas de enlace se hacen en diferentes canales de frecuencias y a distintas velocidades de datos, la selección de la velocidad de datos está relacionada en proporción a la distancia de alcance, la duración y el consumo de energía del mensaje.

Debido a la tecnología de espectro ensanchado (o SS, spread spectrum), las comunicaciones a distintas velocidades de datos no interfieren con otras comunicaciones en el medio, creando así un juego virtual de canales que incrementan la capacidad de la puerta de enlace.

Las velocidades de datos se encuentran en el rango de 0,30 Kbps a 50 kbps. Para maximizar en forma conjunta la duración de la batería de los dispositivos finales y la capacidad de la red, el servidor central LoRaWAN maneja la velocidad de datos para cada dispositivo de forma individual, por medio de un esquema adaptativo de velocidad de datos (o ADR, adaptive data rate).

### 2.7.2 Sigfox

Sigfox es una red de largo alcance y baja velocidad que permite la comunicación de datos entre dispositivos, sin estar directamente ligado a la cobertura y disponibilidad de una red móvil cecular.

Esta conexión entre equipos es posible gracias a su tecnología de radio de banda ultra estrecha. Es energéticamente bastante eficiente y utiliza bandas de frecuencia libres de licencia, disponibles en todo el mundo, como las bandas ISM, esto le obliga en consecuencia a no poder ocupar un canal de comunicación por más de un cierto porcentaje de tiempo conocido como duty cycle, imponiendo una de las limitaciones más fuertes sobre Sigfox, es que el límite en el número de mensajes diarios por dispositivo es de 140 [19].

Desde un punto de vista técnico, como hemos visto Sigfox depende de otra red distinta, basada en 868 MHz. Cada nodo puede cubrir un área de cobertura bastante grande, además, empresas que necesiten mejorar la cobertura en su área pueden instalar un equipo repetidor [19].

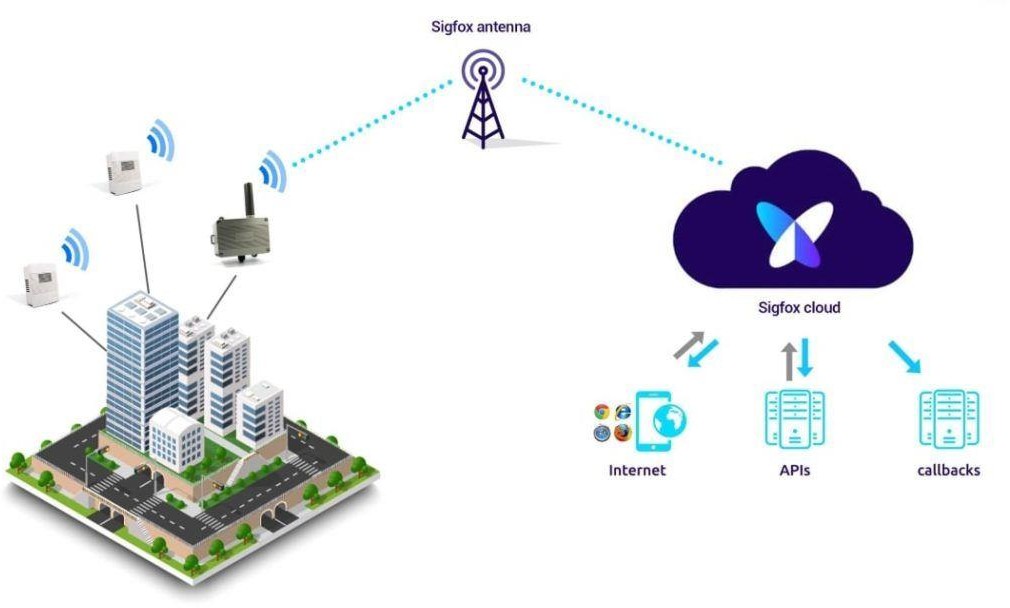
Sigfox tiene un sistema en la nube que desde la interfaz web es posible dar de alta los equipos, que funcionan por un ID único en vez de autenticación por tarjeta SIM como los móviles u otros dispositivos IoT basados en GPRS como se puede ver en la Figura 2.7.2.11.

Figura 2.7.2.12: Sistema en la nube de sigfox [19]

También ofrece una red con buena calidad de servicio que garantiza de manera efectiva bajos volúmenes de datos, pero que puede soportar muchos dispositivos simultáneos como por ejemplo un despliegue de sensores lo cual permite desarrollar aplicaciones independientes con dispositivos IoT propios [19].

## APIs de geolocalización

En esta propuesta de proyecto hablaremos de dos API de geolocalización que son google map y Opend Street Map.

### 2.8.1 Google Maps

La API ofrece una serie de funciones que permiten a los programadores integrar los servicios de Google en sus desarrollos webs (o aplicaciones móviles) de manera más o menos sencilla, sin tener que volver a programar todas las funcionalidades de Google Maps ni tampoco tener que conocer cómo lo han programado.

Desde las APIs se pueden generar mapas con estilos personalizados, crear rutas a partir del tráfico o consultar las imágenes de Google Earth o Street View, entre otras muchas cosas.

La API de Google Maps es un servicio de mapas de Google que brinda diferentes funcionalidades tales como: marcadores, asignación de rutas, trazar sectores dentro de un mapa, entre otras. Ahora nombraremos cuatro prácticas útiles que se pueden hacer con la API.

* Crear una aplicación donde se tenga que marcar una zona dentro del mapa o dar indicaciones para llegar a un destino. Un ejemplo de aplicaciones que hacen uso de este servicio son Uber, Tinder, periscope, etc.
* A través de una app web o móvil la API de Google Maps puede mostrar a los usuarios bares, cafeterías, aeropuertos o tiendas de alimentos cercanos, usando una lista filtrada con los lugares más relevantes de la zona.
* Estimar la duración de un viaje en función de los datos históricos por horario y día de la semana.
* La API de Google Maps les permite a las aplicaciones aplicar un código de colores a las calles principales para reflejar el volumen actual del tráfico en tiempo real.

### 2.8.2 Open Street map

Open Street Map también conocido por sus siglas OSM, es uno de los más populares servicios de mapas online de internet y un proyecto encaminado a la creación, publicación y visualización de mapas para su uso en diferentes plataformas y dispositivos.

Accediendo al sitio, podrás cargar mapas de cualquiera región o zona geográfica. Es un proyecto de código abierto, en el que pueden participar cualquier persona. La principal característica de OSM, es que los mapas son creados y editados por los propios miembros de la comunidad, aportando cada uno su conocimiento. Cualquier persona puede colaborar, siempre que cuente con una conexión a Internet con la que poder subir los mapas editados, a medida que más personas aporten, más se enriquecerá el contenido de dichos mapas.

## Trabajos relacionados

A continuación, se presentan trabajos relacionados con la presente investigación. El motivo para describir dichos trabajos es que tienen enfoques muy semejantes a la solución en la que se centra el estudio y cuentan con funcionalidades y características que se desean contemplar en el desarrollo del mismo. Además de que dan una idea a los autores sobre qué requerimientos se deben satisfacer para obtener una aplicación competitiva y realmente útil para los usuarios, por medio del análisis de las funciones implementadas y los resultados obtenidos de cada uno.

### 2.9.1 MasterCard Blockchain

Los consumidores siempre han querido que las afirmaciones éticas que las compañías acerca de sus productos sean reales. Mastercard está colaborando con Envisible, una compañía que permite que la visibilidad de la cadena de suministro en para así brindar un monitoreo constante y mejorar sus procesos de transporte de alimentos.

El sistema de trazabilidad Wholechain de Envisible es impulsado por una solución basada en blockchain de Mastercard y utilizada por Topco Associates, LLC, una cooperativa de alimentos líder de los Estados Unidos, para ayudar a los propietarios de supermercados a rastrear y resaltar el origen de los mariscos. Topco está trabajando con sus cadenas de supermercados miembros, comenzando con Food City, para probar el uso de la tecnología para proporcionar una mejor línea de visión en buen abastecimiento y el cumplimiento ambiental de la selección de mariscos que se vende en sus tiendas. Las primeras de varias especies que se rastreará será el salmón, el bacalao y el camarón.

Con más de 100 patentes de blockchain presentadas, Mastercard es el número tres a nivel mundial entre los principales innovadores de blockchain. La compañía ha construido una red de blockchain autorizada que está integrada con su red global y proporciona un libro de contabilidad seguro a prueba de manipulaciones, alto rendimiento de transacciones, seguridad mejorada, privacidad de transacciones y soporte para múltiples casos de uso con una sola implementación.

### 2.9.1 IBM FoodTrust

IBM Food Trust es la primera solución de seguridad alimentaria blockchain que permite a los socios de transacciones compartir información alimentaria de forma segura y segura, creando una cadena de suministro mundial de alimentos más transparente y confiable.

La red de IBM Food Trust permite el rápido rastreo de los alimentos, es transparente, segura, obtener visibilidad en el mapa, consultar la ubicación o el estado, y verificar la credibilidad o la seguridad. El módulo Trace permite una gestión eficaz y garantiza la seguridad de los alimentos a través de todo el sistema alimentario.

La red de IBM prueba su sostenibilidad mediante la gestión segura de los certificados en toda la cadena de suministro y con acceso instantáneo a los registros y documentos digitalizados y también permite compartir o ver datos mejorando la confianza y reduciendo el fraude y generando fiabilidad.

Con esta tecnología se puede obtener más información de valor sobre las eficiencias de la cadena de suministro. Analiza en segundos dónde y cómo el producto fresco está más tiempo con el flujo de inventario, el tiempo medio de permanencia, el tiempo transcurrido desde la cosecha.

# 3.Marco Aplicativo

En este capítulo se definen las metodologías usadas a nivel de investigación, diseño y programación de la aplicación descrita en este Trabajo Especial de Grado, así como las herramientas de hardware y software utilizadas para su desarrollo.

## 3.1. Metodología General de Trabajo

A continuación, se describe en detalle la metodología o entorno de trabajo SCRUM, las fases que lo comprenden y cómo estas se adaptan a un proceso de desarrollo de software al proveer un entorno de trabajo ágil, flexible, iterativo e incremental.

### 3.1.1 SCRUM

Scrum es un marco de gestión de proyectos de metodología ágil que ayuda a los equipos a estructurar y gestionar el trabajo mediante un conjunto de valores, principios y prácticas. Al igual que un equipo de rugby (de donde proviene su nombre) cuando entrena para un gran partido, scrum anima a los equipos a aprender a través de las experiencias, a autoorganizarse mientras aborda un problema y a reflexionar sobre sus victorias y derrotas para mejorar continuamente [16].

Aunque son los equipos de desarrollo de software los que utilizan con mayor frecuencia este tipo de scrum, sus principios y lecciones se pueden aplicar a todo tipo de trabajo en equipo. Esta es una de las razones por las que es tan popular. Aunque se considera a menudo un marco de gestión de proyectos ágil, scrum incluye un conjunto de reuniones, herramientas y funciones que, de forma coordinada, ayudan a los equipos a estructurar y gestionar su trabajo [16].

#### 3.1.1.1 Roles principales de SCRUM

* **Product Owner**: El Product Owner o Dueño del Producto, es la persona encargada de lograr

el máximo valor del negocio para el producto o servicio a desarrollar en el proyecto, por ello,

se requiere que articule correctamente los requerimientos del producto, representando así la

voz del cliente dentro del equipo de trabajo [16].

* **SCRUM Master**: Es un facilitador que se asegura que el SCRUM team o equipo de desarrollo

pueda laborar en un entorno que los dirija a la conclusión exitosa del proyecto, siguiendo los

procesos definidos en el entorno SCRUM [16].

* **SCRUM Team**: El equipo SCRUM es el grupo de personas responsables de entender los

requerimientos especificados por el Product Owner y crear o desarrollar los entregables del

proyecto que serán mostrados al cliente al final de cada Sprint o incremento del proyecto [16].

#### 3.1.1.2. Time Boxes de SCRUM

* **Sprint**: Un Sprint es una iteración cuya duración varía entre 1 y 6 semanas, en el que el Scrum

Master y el Scrum Team trabajan en conjunto para convertir los requerimientos del Product

Backlog en funcionalidades concretas del producto. Durante los Sprints se desarrollan

incrementos del producto, es decir, todas las funcionalidades especificadas en el Sprint

Backlog para el avance en la funcionalidad del producto final [16].

* **Reuniones Diarias**: Son reuniones cortas cuya duración varía entre 15 y 30 minutos, en el

que los miembros del equipo se reúnen para reportar el progreso del proyecto [16].

* **Reunión de Planificación de Sprint**: Esta reunión suele llevarse a cabo antes de cada Sprint,

para cumplir con los procesos SCRUM destinados a la asignación de Historias de Usuario,

identificación de tareas, estimación de tareas y creación del Sprint Backlog. Suele durar 8

horas si el Sprint está planificado para extenderse durante un mes [16].

* **Reunión de Revisión de Sprint**: Esta reunión cumple con el fin de validar los resultados del

Sprint, y en ella se presentan los entregables planificados para dicho Sprint al Product Owner.

El Product Owner realiza la revisión del incremento de producto, verifica si cumple con los

criterios de aceptación deseados y acepta o rechaza las historias de usuario completadas [16].

* **Reunión de Retrospectiva de Sprint**: Durante esta reunión, el Scrum Team revisa y

reflexiona sobre las actividades del Sprint previo: Procesos seguidos, herramientas

empleadas, mecanismos de colaboración y comunicación, y otros aspectos relevantes al

proyecto. El equipo discute cuales actividades resultaron exitosas en el Sprint anterior y

cuáles no, para realizar las mejoras correspondientes en los Sprints siguientes [16].

#### 3.1.1.3 Artefactos

Los artefactos del scrum ágil son información que un equipo de scrum y las partes interesadas utilizan para detallar el producto en desarrollo, las acciones para producirlo y las tareas realizadas durante el proyecto. Estos artefactos ofrecen metadatos que dan una idea del rendimiento de un sprint y son los siguientes:

* **Backlog del producto**: es una lista de nuevas funciones, mejoras, correcciones de errores, tareas o requisitos de trabajo necesarios para crear un producto. Se obtiene a partir de fuentes como la atención al cliente, los análisis de la competencia, las demandas del mercado y los análisis empresariales en general, el backlog del producto es un artefacto “vivo”, en el sentido de que se actualiza a medida que se dispone de nueva información [16].
* **Backlog de sprint**: El backlog de sprint es un conjunto de tareas del backlog del producto que se han impulsado para desarrollarse durante el siguiente incremento del producto. Los equipos de desarrollo crean los backlogs de sprints para planificar las entregas de cara a los futuros incrementos y detallar el trabajo necesario para concebir el incremento.

Para generar los backlogs de sprints, se selecciona una tarea del backlog del producto y se divide dicha tarea en elementos de sprint más pequeños y procesables. Pongamos a modo de ejemplo una tarea como “crear una página para el carrito de la compra”, que precisa de muchas subtareas de diseño y desarrollo. El backlog del producto contiene la tarea principal, mientras que las complementarias, como “crear una maqueta de diseño visual del carrito de la compra” o “programar las sesiones del carrito de la compra”, se encuentran en el backlog de sprint [16].

* **Incrementos:** Un incremento del producto es la entrega al cliente que tiene lugar al completar las tareas del backlog del producto durante un sprint. También incluye los incrementos de todos los sprints anteriores.

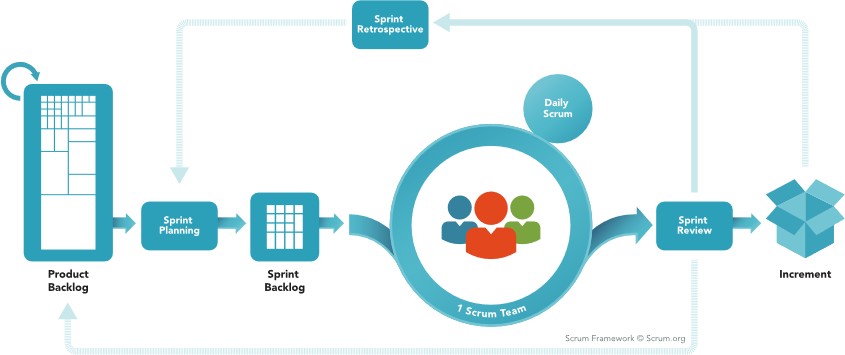


Figura 3.1.1.1: Estructura completa de un sprint en SCRUM [16]

## 3.2. Modificaciones al entorno de trabajo SCRUM

Para el desarrollo del presente aplicativo de cadena de suministro con contrato inteligente es realizó una adaptación del entorno de trabajo SCRUM para un equipo de trabajo sumamente pequeño. Las modificaciones se listan a continuación:

* **Roles**: El rol de Cliente y Product Owner está ejercido por el tutor principal del present e trabajo, en tanto que el SCRUM Team estará compuesto por un solo desarrollador, quien elabora este Trabajo Especial de Grado.
* **Artefactos**: Los artefactos a generar son los mismos requeridos por SCRUM, es decir, un Product Backlog, en el que se describen las funcionalidades necesarias para el producto final, una serie de Sprint Backlogs por Sprint, donde se describen una serie de tareas para completar cada una de las funcionalidades de acuerdo a su distribución por Sprint y los entregables generados al final de cada Sprint, que contarán con su debida documentación y código.
* **Time Boxes**: La duración de cada Sprint se estima de 1 a 4 semanas aproximadamente para el desarrollo de cada una de las funcionalidades descritas en el Product Backlog y desglosadas

en cada Sprint Backlog. Finalmente, la planificación queda estimada para el Product Backlog

cómo se describe en la tabla 3.2.3 a continuación:

|  |  |  |
| --- | --- | --- |
| Sprint | Tarea | Duración |
| 1 | * Levantamientos de requerimientos * Diseño de la solución | 1 semana |
| 2 | * Instalación y configuración de nodo de ipfs * Pruebas de movimientos de archivos con el nodo ipfs * Desarrollo de contrato inteligente en remix para almacenamiento de archivos de IPFS * Pruebas de llamada al contrato inteligente en remix | 2 semanas |
| 3 | * Instalación y confirmación de Blockchain local ganache * Desarrollo del motor del servidor API-RESful mediante flask * Integración del motor con IPFS y el contrato inteligente * Pruebas de integración entre el motor con IPFS y el contrato inteligente usando ganache como blockchain local | 2 semanas |
| 4 | * Desarrollo de interfaz de usuario web usando la librería de flask jinja2 * Desarrollo de simulador de sensores para realizar llamadas al motor para generar inserciones al contrato inteligente. | 3 semanas |
| 5 | * Deployment del contrato inteligente en la testnet sepolia de Ethereum. * Integración entre el simulador de sensores, el motor del server y la interfaz gráfica. * Pruebas funcionales | 4 semanas |

Tabla 3.2.1: tabla de sprints.

## 3.3 Tecnologías utilizadas

En la presente sección y apoyándose en lo explicado a lo largo del Capítulo 2, se describe brevemente las tecnologías y herramientas de desarrollo utilizadas para llevar a cabo las aplicaciones en las que se centra la presente investigación:

* **Truffle**: entorno de prueba para blockchain locales y deployment de contratos inteligentes en las cadenas públicas de Ethereum.
* **Ganache**: es un constructor rápido de una cadena de bloques personal de Ethereum que puede usar para ejecutar pruebas, comandos e inspeccionar el estado de la cadena.
* **Web3.py**: es una colección de librerías que nos permite interactuar con los clientes Ethereum mencionados anteriormente, ya sea de forma local (teniendo el cliente en nuestro propio ordenador) o de forma remota (estando el cliente instalado en otro ordenador) usando los protocolos HTTP o IPC. Web3 nos permite compilar, desplegar e interactuar con nuestros propios contratos inteligentes.
* **API Infura**: proporciona acceso instantáneo a través de HTTPS y WebSockets a las redes Ethereum e IPFS. Facilitando el desarrolla de en el marco de trabajo web3.py. También es un clúster de nodos de Ethereum alojado que permite a sus usuarios ejecutar su aplicación sin necesidad de que configuren su propio nodo o billetera de Ethereum.
* **IPFS**: El Sistema de archivos interplanetario es un protocolo y una red diseñados para crear un método p2p direccionable por contenido para almacenar y compartir hipermedia en un sistema de archivos distribuido.
* **Flask**: es un framework minimalista escrito en Python que permite crear aplicaciones web rápidamente y con un mínimo número de líneas de código.
* **HTML:** siglas en inglés de HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web
* **Bootstrap**: es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.
* **Solidity**: lenguaje de programación orientado a objetos para escribir contratos inteligentes.​ Se utiliza para implementar contratos inteligentes ​ en varias plataformas blockchain, la más destacada, Ethereum.
* **Librería de OpenStreetMap**: Librería que permite implementar el motor de búsqueda y todos los assets de OpenStreetMap para desarrollo de un mapa en una interfaz gráfica.
* **Remix**: entorno de desarrollo de contratos inteligentes en solidity.
* **Pycharm**: es un entorno de desarrollo integrado utilizado en programación informática, concretamente para el lenguaje de programación Python.
* **Sepolia testnet de Ethereum**: testnet de la blockchain de Ethereum con todas sus propiedades.

# 4. Diseño de la aplicación

Para la aplicación de cadena de suministro se propone una infraestrucctura modular para facilitar el desarrollo de los componenetes que la componen de manera individual como lo muestra en la Figura 4.1.

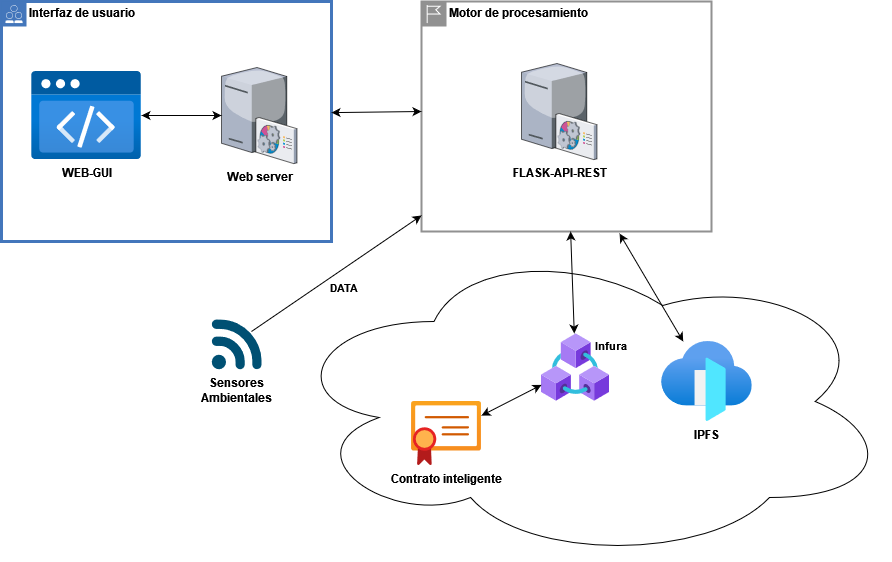


Figura 4.1: Módulos de la solución

Se desarrollaron 4 módulos y 2 submodulos que serán explicados a continuación:

1. **Interfaz de usuario**: consiste en una Web-GUI que está conectada a un web server mediante HTTP. El web server realiza peticiones POST al servidor FLASK-API-REST para recibir la información procesada por este y luego renderisar las vista correspondientes con la información enviada por el servidor FLASK-API-REST en un browser web.
2. **Módulo de procesamiento de información:** este módulo se encarga de recibir peticiones de consulta de información solicitada por el web server y peticiones de carga de información solicitadas por el módulo de sensores, además se encarga de guardar la información en el submódulo de IPFS que generara un hash que será guardado en el contrato inteligente para futuras llamadas del mismo. Está conformado por 2 submódulos de apoyo y estos son:
   1. **Servidor FLASK-API-REST**: Contiene la interfaz de comunicación Infura que permita la comunicación con la blockchain de Ethereum y web3 que se encarga de realizar las llamadas al contrato y el ipfsApi que permite la integración con IPFS para poder generar Hashes de archivos de ipfs .
   2. **IPFS**: módulo que permite la conexión con IPFS hacia el servido FLASK-API-REST atreves de la interfaz de comunicación ipfsApi que permite utilizar todos los comando de manipulación de archivos IPFS.
3. **Módulo de contrato inteligente**: Implementa un contrato inteligente que será ejecutado en la testnet de Ethereum. El contrato tendrá definido las reglas para almacenamiento de hashes enviados por el servidor FLASK-API-REST donde cada dirección pública estará asociada a una ruta de una carga de comida que estará representada por un hash de archivo IPFS y donde estas mismas podrán ser consultadas dada la mismas dirección pública asociada a esa ruta.
4. **Módulo de sensores ambientales**: este módulo simula sensores de latitud, longitud, humedad y temperatura en un script de Python donde genera datos semiRandom que luego serán enviados al servidor FLASK-API-REST mediante HTTP para su posterior procesado.

## Implementación de la solución

#### 4.1.1 Modulo de sensores

### Script programado en Python que simula sensores ambientales y GPS, este módulo enviara un json al servidor FLASK-API-REST mediante un HTTP POST request como lo muestra en la figura 4.1.1.1:

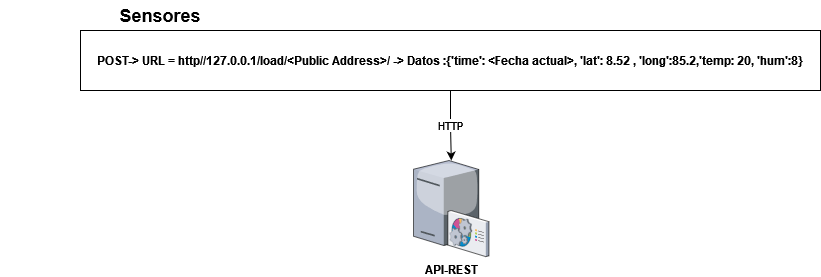


Figura 4.1: Flujo de datos enviados al servidor API-REST

Se enviarán 13 veces en un intervalo de 15 segundos un payload de datos de tiempo, latitud, longitud, humedad y temperatura como lo muestra en la figura 4.1.1.1 que son generados por las funciones presentadas a continuación:

* **getLat(<int>)**: recibe un contador que sirve para recorrer un arreglo de latitudes devolviendo un string para construir el payload así como lo muestra la figura 4.1.1.2.

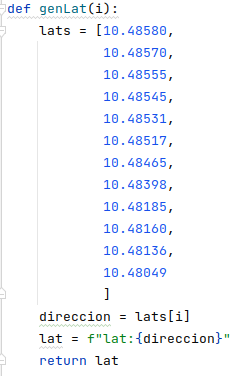


Figura 4.1.1.2: función getLat

* **getLong(<int>)**:recibe un contador que sirve para recorrer un arreglo de longitudes devolviendo un string para construir el payload así como lo muestra la figura 4.1.1.3

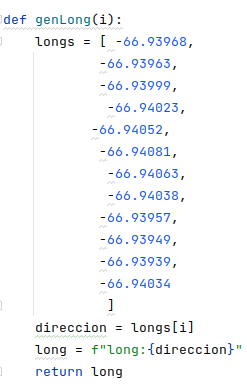


Figura 4.1.1.3: función getLong

* **genTemp()**: genera un valor pseudoaleatorio de un numero entre 0 y 100 para simular valores de temperatura como se muestra en la figura 4.1.1.4.

A black and white text

Description automatically generated

Figura 4.1.1.4: función genTemp

* **genHum**: genera un valor pseudoaleatorio de un numero entre 0 y 100 para simular valores de humedad como se muestra en la figura 4.1.1.5.

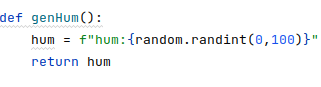


Figura 4.1.1.5: Función genHum

* **Main**: función que se va a ejecutar 13 veces que son la cantidad de puntos que están en una ruta preestablecida en las funciones anteriores donde cada punto se le generaran los datos de latitud, longitud, humedad y temperatura correspondientes para luego construir un json y ser enviado por un método HTTP POST con lo muestra en la figura 4.1.1.6.



Figura 4.1.1.6: Función main

#### 4.1.2 Modulo de procesamiento

Como se mostro en la figura 4.1, consta de submódulos API-REST desarrollado en Python. Hace uso de las librerías flask y flask-restfull para poder recibir peticiones GET y POST, integrado con la librería web3.py que permite realizar consultas al contrato inteligente por medio de la integración con la API de infura. El submódulo de IPFS que hace uso de la librería ipfsApi para poder utilizar comandos de IPFS como código en Python para la creación y manipulación de archivos ipfs que consisten en un hash único que luego será enviado al contrato para ser almacenado. A continuación, se van a detallar el funcionamiento de los distintos end-points en la tabla 4.1.2.1:

|  |  |  |  |
| --- | --- | --- | --- |
| **Recurso** | **Acción** | **Método** | **Datos enviados** |
| /load | Recibe el payload del modulo de sensores para poder cargar la información en IPFS y luego guardar ese hash de IPFS en el contrato inteligente | POST | Mensaje:  ‘Listo <Hash de ipfs de 32 byte>’ |
| /hist | Hace una llamada al contrato que devuelve el hash del archivo ipfs y luego manipula esa información para devolver un arreglo de json que contiene la información de GPS y datos ambientales dada por los sensores | POST | [  {  Time:”…..”  Lat:”……..”  Long:”…….”  Temp:”…….”  Hum: “…….”  },  {},{},{}  ] |

Tabla 4.1.2.1: Endpoints API-REST