# BlockChain-As-A-Service: Análisis y validación

Trabajo Final de Grado (TFG)

Autor: Jesús Alfaro

Director: Jaime Delgado

Fecha: 22/04/2021


# ÍNDICE

# 1. Contexto

## 1.1. Introducción

La fiabilidad digital es algo que en los últimos años, a medida que la tecnología iba cobrando fuerza y calando cada vez más en la sociedad, ha tomado importancia en diferentes ámbitos. Dentro de las tecnologías que pueden aportar más fiabilidad y que puede aplicarse a prácticamente todo se encuentra el blockchain.

La tecnología blockchain tiene tres importantes características que pueden ser muy útiles en diferentes ámbitos laborales como puede ser la información distribuida (P2P) que hace más fiable la información, la transparencia de la información y la seguridad criptográfica.
Además hay tres grupos que dividen las ofertas blockchain que son las redes privadas, las públicas y las de consorcio (con permisos).
Las redes privadas y con permisos son a las que hará referencia este proyecto. La principal diferencia es que en una red privada no puede acceder cualquier persona y en una pública sí. Esto es importante ya que las empresas ofrecen como servicio este tipo de red ya que su gestión es más compleja al haber más niveles de autorización y por ende más seguridad.
En la actualidad existen distintos frameworks blockchain que son utilizados por diferentes compañías tecnológicas para dar lugar a plataformas blockchain que ofrecen el servicio (BaaS) blockchain-as-a-service. Proporcionando hardware y software a sus clientes para poder desarrollar y desplegar aplicaciones distribuidas con la tecnología blockchain.
Este proyecto se centrará en los dos tipos de framework que utilizan tres empresas diferentes que ofrecen BaaS. Se hará una comparativa y se validará los resultados implementando una aplicación con uno de los framework.
El proyecto está delimitado por el marco de la FIB, en concreto con la especialización de Tecnologías de la Información (TI) en cuanto a la identificación de diferentes tecnologías existentes, entenderlas e integrarlas para crear nuevas aplicaciones.

## 1.2. Trasfondo

Los framework de blockchain integran varias tecnologías que ya se utilizaban hace años y otras que han aparecido o tomado fuerza recientemente. Hay ciertos elementos que pueden resultar de interés definir.

**Smart contract o contrato inteligente**: Un smart contract es un programa informático o un protocolo de transacción que está destinado a ejecutar, controlar o documentar automáticamente eventos y acciones legalmente relevantes de acuerdo con los términos de un contrato o acuerdo.
Es decir, un smart contract debe ser aprobado por todos los nodos de la red blockchain antes de poder interactuar con el mismo.

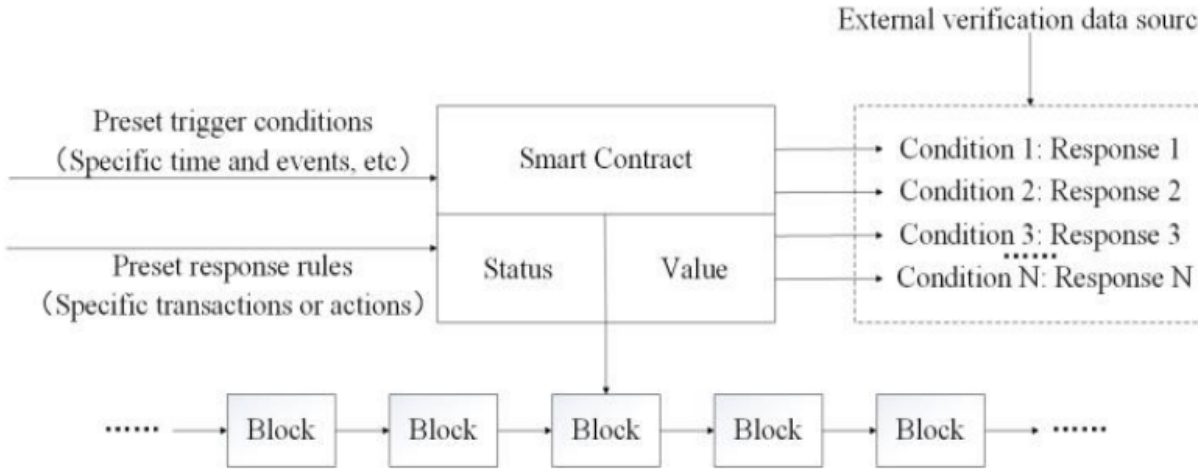


Figura 1. Mecanismo operativo de los contratos inteligentes. [3]

El smart contract encapsula un número de estados y reglas (funciones) de transición predefinidos, escenarios que desencadenan la ejecución del contrato (como en un momento dado o un evento determinado ocurre), dando respuestas en un escenario particular, etc.
Blockchain monitorea el estado en tiempo real de los contratos y ejecuta el contrato después de cierto desencadenante que indican las condiciones. [1] [2] [4]

**Capas de la tecnología blockchain:** Normalmente un sistema de cadena de bloques consiste de seis capas.

**Data layer:** Esta capa incluye los datos subyacentes bloques, mensajes cifrados relacionados y marca de tiempo, etc.

**Network layer**: El sistema blockchain generalmente adopta el protocolo P2P que está completamente distribuido y puede tolerar un solo punto de fallas (SPoF).

**Consensus layer**: La capa de consenso encapsula varios tipos de protocolos de consenso. Esto es debido a que la cadena de bloques descentralizada se gestiona de forma conjunta y es mantenida por múltiples partes. El consenso se basa en validar un bloque de datos (a través de un puzzle sha256), se compite por validar un bloque y el primer nodo o minero que lo consiga transmite el bloque a todos los demás y estos confirman su

validez. Si un bloque está validado otros mineros encadenan ese bloque a su cadena.

**Incentive layer**: Los mecanismos compatibles con incentivos deben ser diseñados, de modo que el comportamiento racional individual de los nodos de consenso para maximizar sus propias ganancias puede alinearse de forma incentivada con el objetivo general de garantizar la seguridad y eficacia del ecosistema blockchain descentralizado.

**Contract layer**: La capa de contrato encapsula varios tipos de códigos de secuencia de comandos, algoritmos y contratos inteligentes sofisticados, y por lo tanto es la base para programación flexible y manipulación de sistemas blockchain.

**Application layer**: La principal aplicación en el sistema Bitcoin es transacciones de moneda digital. Para la plataforma Ethereum, además de las transacciones de divisas, también admite aplicaciones descentralizadas (Dapp). [3]

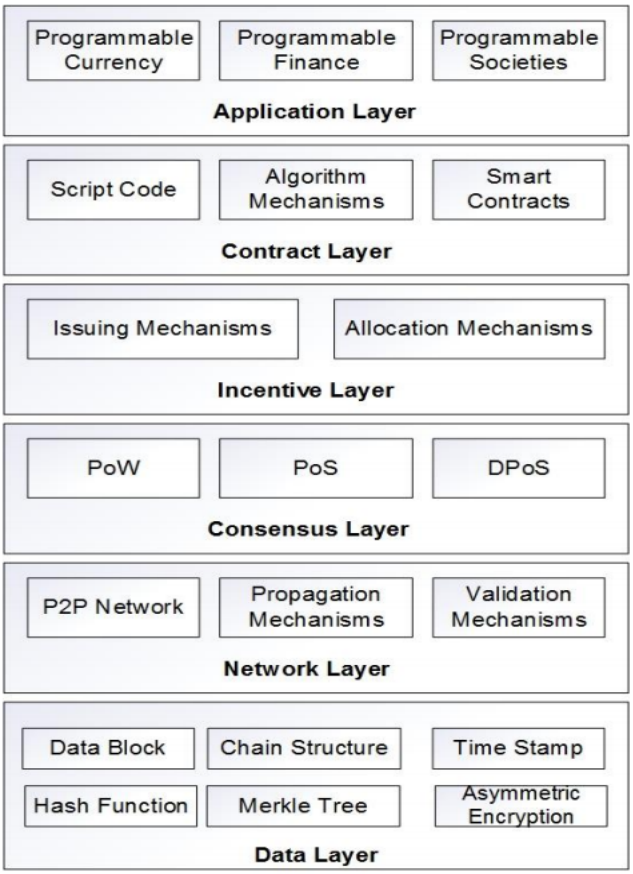


Figura 2. Framework básico de blockchain [3]

**Paradigma Blockchain:** Se basa en la distributed ledger technology (DLT) donde cada participante tiene acceso a un libro mayor (ledger). La idea de tener un libro mayor abierto y universalmente accesible (en caso red pública) nació con Bitcoin, y

el sistema proporcionó la primera solución al problema de establecer confianza en un entorno inseguro sin depender de un tercero. [6] [7]

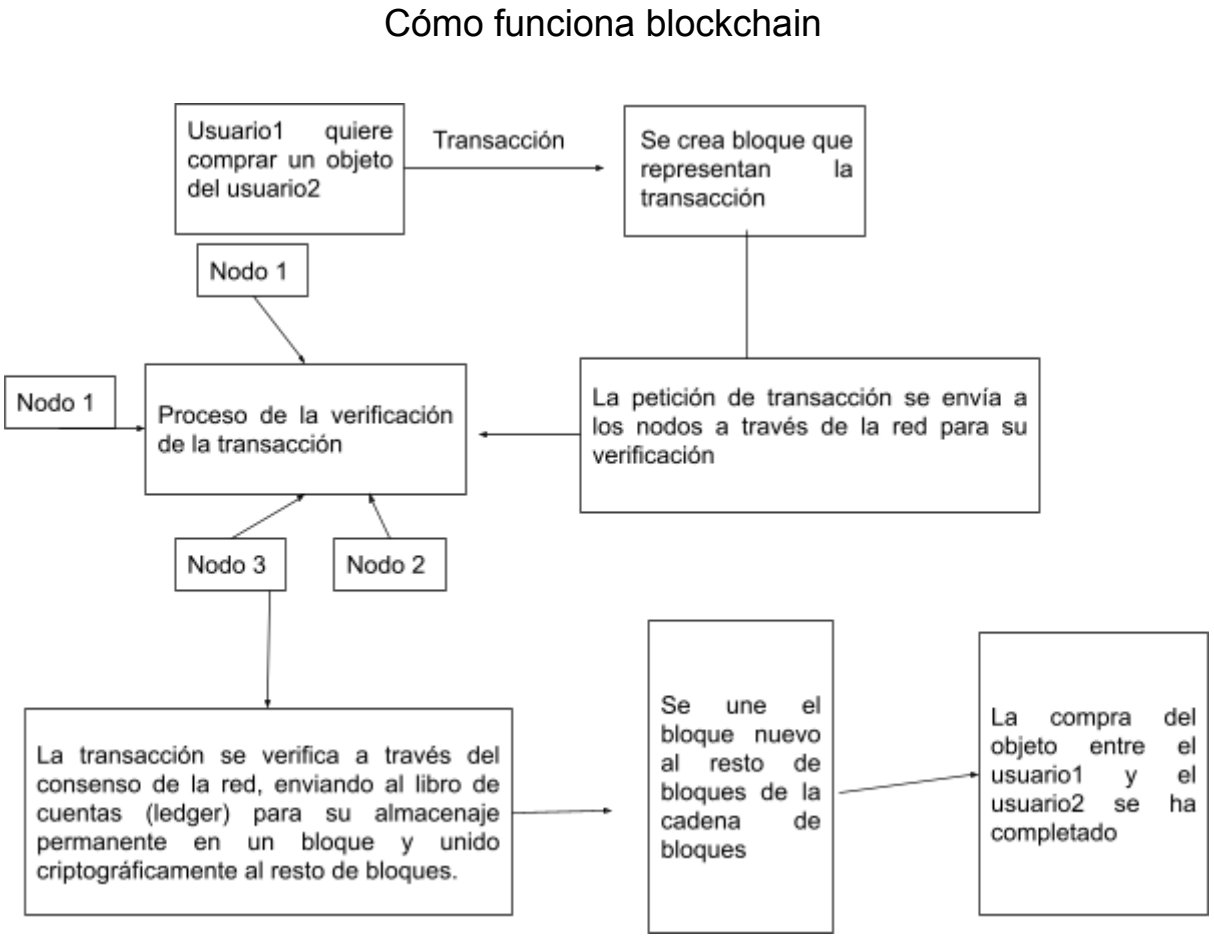


Figura 3. Representación conceptual del paradigma de cadena de bloques. [5]

**Solidity**: Solidity es un lenguaje de alto nivel orientado a contratos inteligentes. Su sintaxis es similar a la de JavaScript y está enfocado específicamente a la Máquina Virtual de Ethereum (EVM).

Solidity está tipado de manera estática y acepta, entre otras cosas, herencias, librerías y tipos complejos definidos por el usuario. [8]

## 1.3. Objetivo del proyecto

Demostrar qué framework blockchain es el mejor de dos opciones muy populares mediante una comparativa sobre diferentes aspectos como capacidades,

infraestructura, implementación, entre otros. De esta manera el proyecto ayudará a identificar de manera rápida el mejor servicio a utilizar por organizaciones o particulares.
Además se validará la información con el desarrollo de una aplicación que se comunicará con un smart contract desplegado en la red blockchain de una de las opciones.

## 1.4. Actores implicados

### Desarrollador

El desarrollo de la aplicación distribuida se basará en una interfaz web implementada con Vue.js.
El contrato inteligente se basará en activos (objetos) iniciales y funciones predeterminadas que servirán para realizar consultas o modificaciones de los activos.
Búsqueda de información sobre la programación necesaria, posterior testeo y corrección de errores.
Búsqueda de información sobre los dos frameworks, compararlos y documentarlos.

### Director y tutor del proyecto

El director del proyecto es Jaime Delgado y su papel es el de supervisar el desarrollo del proyecto y de si se alcanzan los objetivos estipulados. También puede guiar al desarrollador y resolver dudas sobre el proyecto.
Joan Sardà es el director de GEP del proyecto y su papel es el de corregir y guiar al estudiante mediante rúbricas durante la fase gestión del proyecto.

### Interesados

El público al que va dirigido la información de la comparativa son empresas, organizaciones, o bien particulares interesados en utilizar esta tecnología. La información estará estructurada y será fácil de entender.
La aplicación, además de validar la información, tiene una temática orientada a ayudar a grupos de trabajo, se basa en la organización y gestión de tareas.
Por lo que centros educativos pueden estar interesados en optar esta herramienta para fomentar el uso de la tecnología y el trabajo grupal.
Otros interesados pueden ser empresas que la quieran poner a disposición de sus empleados o bien ofreciendolo como una plataforma para diferentes usuarios de la red.

# 2. Justificación

La tecnología blockchain ha cogido fuerza e importancia en los últimos años y esto puede llevar a que haya ciertos interesados en conocer más sobre el tema. Como una manera de ayudar y dar a conocer más sobre las características de diferentes frameworks blockchain, utilizaré aquellos que son utilizados por grandes empresas que se dedican a dar servicios tecnológicos. El llamado blockchain-as-a-service.

Se hará una comparación entre diferentes frameworks dejando en claro qué ventajas y desventajas existen entre ellas y sacar en claro cuál es la mejor opción según qué necesidades.

Posteriormente, como una forma de validar la información, se pasará a desarrollar una aplicación con un contrato inteligente que se integre en el despliegue de una red blockchain de una de las opciones comparadas y también se presentará una descripción del funcionamiento de la red blockchain, identificando los componentes característicos que intervienen en la ejecución de la aplicación distribuida.

Existen en internet diversas comparativas entre diversas empresas que ofrecen el blockchain as a service pero en lo que se centrará este proyecto será en un enfoque diferente al desarrollar una aplicación y describir los pasos necesarios para llevar a cabo el despliegue de la red de Hyperledger Fabric.

La aplicación irá destinada hacia grupos de trabajo ya que su función será gestionar las fechas de entrega que los participantes del grupo hayan establecido. Existen diversas aplicaciones web para ayudar al trabajo grupal como puede ser Monday.com pero en este caso lo que se intenta hacer es automatizar el papel del responsable del grupo que se encarga de controlar el progreso del resto de los integrantes del grupo al sustituirlo con un contrato inteligente y los nodos que forman parte de la red.

# 3. Alcance del proyecto

Primeramente se llevará a cabo una comparativa entre los framework que utilizan IBM blockchain platform, Microsoft Azure blockchain y Oracle blockchain platform en forma de blockchain-as-a-service.
Los frameworks correspondientes son Hyperledger Fabric [9] para IBM [11] y Oracle [12], y Ethereum Quorum [10] para Microsoft [13].
Se comparará la infraestructura, teniendo en cuenta cuál es más sencilla de entender, cuál está mejor documentada y qué funcionalidades ofrece, el rendimiento, los usos que pueden tener o qué empresas están utilizando dichos frameworks.
Para poner en práctica y validar lo que se ha visto, se desarrollará una simple aplicación de contratos inteligentes con Javascript, se desplegará en una configuración de red con uno de los framework y con una interfaz de usuario desarrollada con Vue.js.

## 3.1. Comparativa

La comparativa tiene como objetivo sacar en claro qué framework es mejor según ciertas características, detalladas más adelante, o bien las diferencias que existan. Como la infraestructura de cada una de las dos, su funcionamiento, etc.
Al final se recogerá la información en un cuadro comparativo que tendrá como parámetros las características de la comparación, pudiendo sacar mejores conclusiones al contrastar un resumen de los resultados.
Las conclusiones dejarán claro cuál de las dos opciones es mejor o resulta más beneficiosa para trabajar y de esta manera ayudar en la elección de terceros.

## 3.2. Aplicación y  Herramientas de desarrollo

Como se ha comentado anteriormente, la aplicación tendrá el fin de validar la información recopilada. La descripción de la aplicación descentralizada se basa en tareas que representarán los activos/objetos de las transacciones, es decir, cada nodo guardará el estado de cada una de las tareas y su aprobación también deberá ser validada por un mínimo de peers (mayoría) antes de confirmar el cambio en el resto de nodos (sus bases de datos).
Se utilizará el software libre Hyperledger Fabric que contiene diferentes binarios, librerías, SDK, ejemplos de funcionamiento y configuración. Los diferentes componentes como los nodos/peers se crearán con contenedores Docker.
Se describirán los componentes principales de la implementación, empezando por la configuración de la red, el papel del contrato inteligente, el funcionamiento de la red cuando los usuarios interactúan con la interfaz web.

## 3.3. Ejecución de la aplicación

La ejecución de la aplicación consistirá en una red que tendrá como hardware contenedores docker que simularán ser otros peer y autoridades de certificación, necesarios para el correcto funcionamiento de la red.
El usuario que será un propietario de tareas en blockchain, podrá interactuar con la web desarrollada con Vue.js en localhost.
Cuando un usuario realice una acción de modificación del estado de sus tareas se podrá observar cómo se valida la transacción y también qué peers/nodos han sido los responsables. Cada usuario únicamente puede modificar sus tareas no las de los demás usuarios, pero sí podrá consultarlas. Cada peer será responsable de almacenar en su base de datos el estado de todas las tareas para de esa forma garantizar la seguridad y validez de los mismos.

## 3.4. Metodología y rigor

Para elaborar el proyecto se dispone de unos (4 meses), por ello, hay que llevar unos ciclos de trabajo cortos y respetarlos lo máximo posible ya que de lo contrario es posible que no dé tiempo a depurar la aplicación. Se hizo una ampliación del tiempo para la presentación de 3 meses ya que se buscaba no ir con prisas en las últimas fases del desarrollo.

Primeramente la fase de búsqueda se centrará en la documentación oficial de los diferentes frameworks para evitar información desactualizada o errónea, además de videos explicativos o ejemplos de código.
El entorno de desarrollo VC Studio será con lo que implementará el contrato inteligente y configure la red. Además el progreso será guardado en Github para evitar pérdida de información y permitir trabajar desde cualquiera máquina.

Una vez creada la aplicación, la puesta en marcha se hará simulando un grupo de tres participantes. Las pruebas consistirán en :

- Uno de los participantes no entregará su trabajo a tiempo y el contrato inteligente indicará que la tarea no ha sido completada a tiempo.
- Se probará a crear una tarea nueva para un usuario siguiendo las funciones y reglas que marca el smart contract. Probando así la lógica del mismo.
- Mirar cómo se adapta la tabla de progresos cuando se hacen las entregas a tiempo.

La herramienta para llevar un buen ritmo de desarrollo será un diagrama de Gantt, en el que se irán marcando los objetivos planteados por semanas.
Como trabajo a medio tiempo por la mañana dedicaré todas las tardes al proyecto. Los fines de semana no se le dedicarán horas a menos que se vea en riesgo superar las fechas establecidas para los objetivos.
Se llevará a cabo una reunión de seguimiento con el tutor cada dos semanas para asegurar que se están cumpliendo los objetivos.

## 3.5. Posibles obstáculos

La configuración de la red será un obstáculo ya que será la primera vez que haga algo parecido y un error de configuración provocará que la red no funcione correctamente. Localizar el error puede ser tedioso ya que habrá que repasar gran parte del código o logs y entenderlo para poder corregir el error, me ayudaré de la información en internet y los foros de ayuda de programación.
A parte de la configuración de la red, también estará la programación del contrato inteligente que deberá seguir un lenguaje de programación específico, en Hyperledger Fabric se puede utilizar Javascript, Java, Typescript y Go. Para Ethereum Quorum se utiliza Solidity. En ambos casos habrá que, al igual que en la configuración de la red, entender el funcionamiento de un contrato en el ecosistema blockchain para programar adecuadamente su funcionamiento.

Es posible que la aplicación no funcione como debiera por la existencia de bugs. El incorrecto funcionamiento de la aplicación podría ser que no avise a todos los participantes cuando uno de ellos no entregue su trabajo antes del deadline especificado. Que el comportamiento de la aplicación no altere la visualización de las tareas adecuadamente cuando las entregas se realicen a tiempo, entre otros. Estos errores se podrán originar por una mala programación del contrato inteligente o bien por una configuración errónea de la red/servidor.

Todos estos problemas pueden llevar a retrasos en la planificación del proyecto, derivando en más horas de trabajo de las estimadas y por lo tanto peores resultados finales si no se consigue ponerse al día con el calendario.

# 4. Planificación temporal

La duración del TFG será de cuatro meses y una semana, desde el 14 de septiembre al 22 de enero. Durante este periodo se realizarán una serie de tareas para elaborar correctamente el TFG. Se realizó una extensión del plazo de entrega hasta abril para poder desarrollar el proyecto sin prisa.

## 4.1. Descripción de tareas

Buscar información de los framework será la primera tarea.

Se buscará en la página web oficial de Hyperledger Fabric, la documentación necesaria para desarrollar una aplicación con smart contract y cómo se integra en una red distribuida en la red de bloques.
Se hará lo mismo con Ethereum Quorum, la estimación que se propone es de 10 a 15 horas para cada framework ya que se irá enfocado al servicio de blockchain y a recopilar la mayor cantidad de información, luego en la fase de comparación se puede volver a recoger información que se haya pasado por alto para completar la tarea.

Una vez se haya recogido la información de los dos frameworks a estudiar, se organizará, identificando las semejanzas y diferencias que existan entre los dos. Luego se procederá a hacer una comparativa sacando en claro cuál es la mejor opción para según qué necesidades queramos complacer. De esa manera se opta por una sobrestimación de 25 horas para la fase de comparación. La comparación irá desde infraestructura (seguridad, consenso, componentes, despliegue de red), rendimiento, calidad de servicio y casos de uso.

En principio se desarrollará la aplicación con Hyperledger Fabric pero después de acabar la recopilación y comparativa de información quizá haya un cambio de planes ya que se puede haber identificado una mejor alternativa para el desarrollo. Por ello es importante llevar a cabo una búsqueda de información lo más exhaustiva posible. Aquí se puede apreciar una dependencia del proyecto, ya que sin haber hecho una investigación adecuada no es posible identificar la mejor manera para el desarrollo de la aplicación.

Una vez la comparativa esté finalizada se pasará a planificar el desarrollo de la aplicación, para ello se tendrá que tener preparado todo el software necesario así como todas las dependencias necesarias, como librerías, programas, manuales útiles,etc.

En este momento puede que se deba releer la documentación que se haya recogido para tener claro cuál es el servicio que se elegirá para el desarrollo de la aplicación, de esta manera se hace una sobrestimación de 20 horas para esta tarea y prever posibles cambios.
Una vez se tenga todo lo necesario habrá que familiarizarse con los lenguajes de programación y los códigos de Fabric [14] así que se creará un contrato simple de prueba (prototipo) con el que se podrá identificar problemas que no se tuvieron en cuenta y además agilizar el posterior desarrollo de la aplicación ya que quizá se pueda reutilizar código para la aplicación final, para esta tarea se estima un tiempo de 30h.

El desarrollo de la aplicación será la parte a la que se dedicará más tiempo ya que será en la que surgirán más inconvenientes o dudas en la programación. El lenguaje será javascript pero habrá que utilizar ciertas funciones propias del framework en cuestión, por ello se tendrá que consultar la documentación regularmente, con una sobrestimación de 80h.
El testeo de errores se llevará a cabo en la parte de configuración de red, en la programación del contrato y en el desarrollo de la aplicación web. Se identificarán los errores que se puedan probar por sí solos y luego con la unión de las tres partes se realizará un test completo para identificar posibles bugs de funcionamiento o configuración, la estimación será de unas 40 horas.

Además habrá una entrega al director del proyecto de un informe de seguimiento, para esta tarea se estima unas 15h de preparación.

Se harán reuniones al terminar ciertas fases con el director del proyecto Jaime Delgado para plantear dudas o bien para asegurar que se han cumplido con los objetivos establecidos durante la fase de desarrollo. Aunque se pueden acordar reuniones que no estén en el calendario para consultas, durante la fase de búsqueda se estima que con 10h será suficiente, en cambio con la fase de desarrollo surgirán más dudas por lo que 25h serán más viables .

| Resumen Tareas |
| --- |
| Estudiar ofertas blockchain-as-a-service |
| Compararlas |
| Validar información con aplicación |
| Demostración |

Tabla 1. Resumen de tareas principales

## 4.2. Estimaciones y Gantt

El proyecto tendrá una parte de documentación y por ello hay que respetar el calendario para evitar contratiempos y retrasos que luego se podrán reflejar en la calidad del trabajo final.

En el Gantt se puede ver que el total de horas necesarias son 425h, por lo que dividas por el número de días que se tiene para elaborar el proyecto que son 92, se obtiene el número de horas diarias necesarias para el proyecto, que son 4.30h. No se ha tenido en cuenta los fines de semana.
Para tener un mejor seguimiento de las tareas y del tiempo se utilizará un diagrama de Gantt que se puede encontrar al final del documento.
Las filas del diagrama tienen la lista de tareas, en las columnas se puede encontrar la dependencia de tareas, las semanas y también aparecen las horas que teóricamente se tendrá que invertir en cada tarea, al final salen las horas totales.

La dependencia de tareas funciona de esta manera:

TN donde N es un entero del 0 al 10

TN < T(N+1) significa que T(N+1) depende de TN

Una N menor indica menor dependencia y no se podrá continuar con una tarea si no se ha cumplido las N anteriores.
En el gantt están indicados la TN (dependencias) correspondiente por cada tarea.

Durante la elaboración del proyecto se decidió extender el plazo hasta abril para acabar tanto la parte de desarrollo como la memoria. Como se extendió 3 meses y el trabajo iba suficientemente avanzado se decidió no dedicar tantas horas como en la fase previa de desarrollo. La suma de horas totales es de 500 h, es decir, 75 h extras que se distribuyeron (+40 h documento) y (+35h implementación).

## 4.3. Gestión de los riesgos: Planes alternativos y obstáculos

Con la metodología Agile hay que tener siempre en cuenta que pueden surgir nuevos requisitos o problemas según se vaya avanzando en el proyecto, como por ejemplo, los posibles retrasos en las fechas límite de las tareas y si esto ocurre lo mejor sería aumentar la carga de trabajo en las fases posteriores para intentar cuadrar el calendario.

Una peculiaridad de los smart contract es que son inmutables y por ello cualquier bug que no se localice a tiempo, estará presente en el programa durante toda su ejecución en la cadena de bloques, ya que no se podrá modificar el contrato una vez esté desplegado. Por ello se realizarán tests con ayuda de la librería Chai para

asegurar que las principales funcionalidades de la aplicación funcionan correctamente, ya que serán los problemas más difíciles de identificar y resolver por la falta de experiencia en el tema.

Es posible que la aplicación no funcione como debiera y por ello es recomendable que la planificación del proyecto considere alargar plazos del desarrollo u acortar la duración de otras tareas, por ello se hace una sobrestimación del desarrollo de la aplicación y se estima que unas 80 horas sería suficiente, esto puede verse reflejado en el Gantt.

También se tendrá en cuenta la desviación del tiempo invertido en cada tarea una vez finalice, para ello se hará el cálculo.

Hr_desviacion=Horas_teoricas_tarea - Horas_reales_tarea

Si Hr_desviación > 0 se estará dentro de la planificación y no se tendrá que tomar ninguna medida al respecto.
Si Hr_desviación <= 0 significa que las horas reales invertidas son superiores a las estimadas y se estará incumpliendo la planificación del Gantt. Primero se anotarán el número de horas reales que ha llevado finalizar la actividad y aumentará las horas de trabajo diarios (de 4.30h a 5.30h) hasta cuadrar el horario, pudiendo añadir los fines de semana y anotar qué fines de semana se optó por trabajar en el proyecto para alcanzar la planificación, así como las horas.

Las herramientas que se utilizarán son open source así que no habrá problema con licencias ni nada parecido. Como se ha comentado anteriormente el problema puede ser que haya retraso en las tareas por imprevistos o errores en la fase desarrollo. Eso provocará alteraciones en la planificación inicial que corresponderá a la tarea T7. Este posible incremento en el tiempo se verá reflejado en el recuento de horas reales por actividad y en la desviación.

Si surge algún obstáculo imprevisto que no se pueda resolver, se podrá consultar con el director para que dé indicaciones sobre la mejor manera de llegar a una solución o plan alternativo.

## Herramientas necesarias

Entre las herramientas necesarias para llevar a cabo el proyecto se puede considerar software como:

- Visual Studio Code
- Google drive
- Gmail

- Git
- Github
- IBM blockchain platform
- Oracle
- Microsoft Azure
- Solidity
- Java
- Javascript
- Linux
- Máquina Asus core i7
- Smartphone
- Ethereum
- Hyperledger Fabric
- Vue.js
- Mocha/ Chai
- Visual Paradigm [17]
- StackOverflow [19]

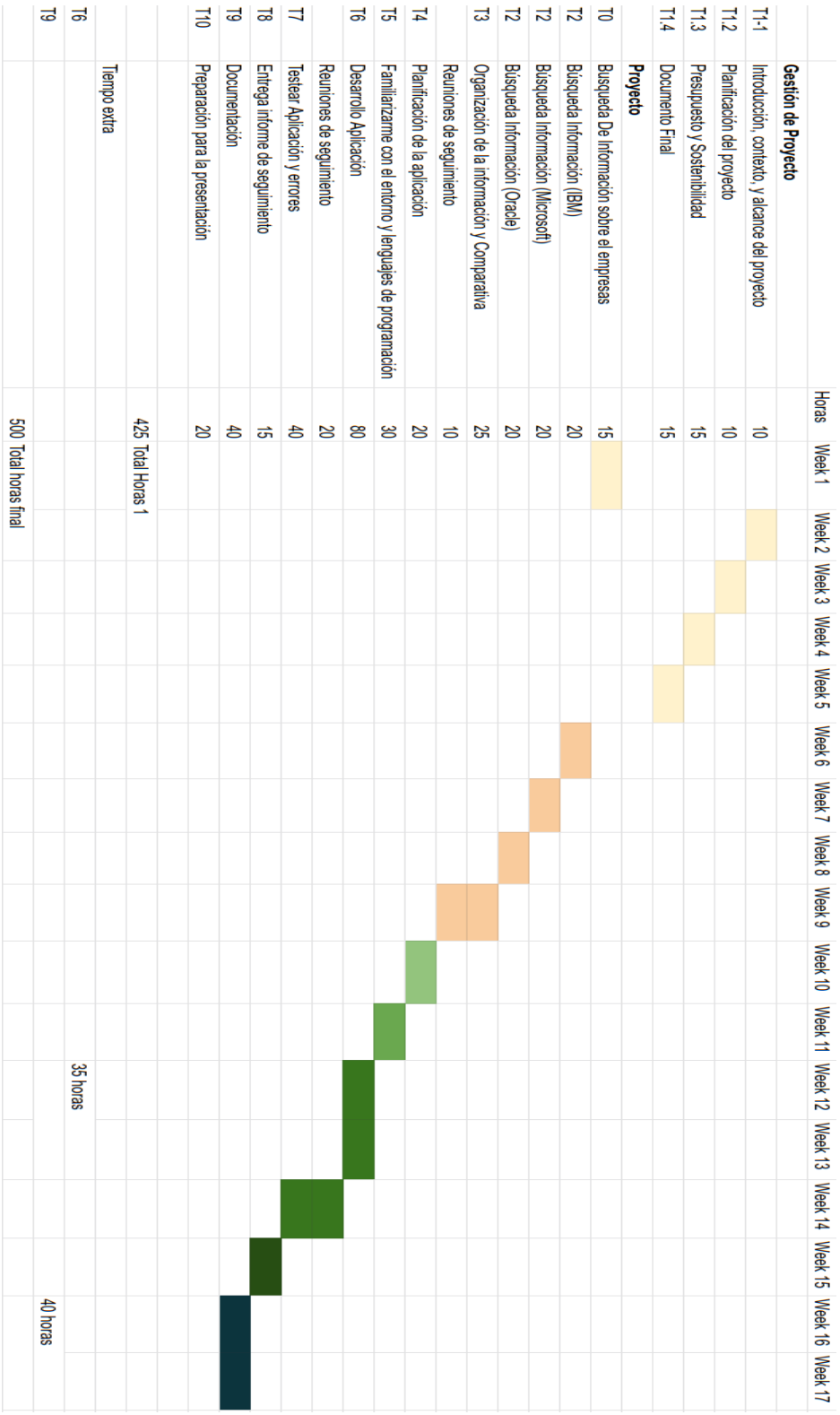


Figura 4. Gantt [Hoja de cálculo, hecho por mi]

# 5. Gestión económica

## 5.1. Identificación de costos y estimación

### RECURSOS HUMANOS

Existen diferentes roles relacionados con las distintas tareas del proyecto.
Hay cinco roles diferentes:
El gestor de proyecto es la persona encargada de que la planificación y el transcurso del proyecto se lleve a cabo correctamente.
El encargado de búsqueda de información es la persona que buscará la información necesaria sobre las empresas y los servicios que ofrecen, también hace la comparación.
Programador quien implementará la aplicación.
Tester quien comprobará el correcto funcionamiento de la aplicación
Escritor técnico es quien escribe todo lo que conlleva el desarrollo, los resultados y conclusiones.

| **Rol** | **Sueldo anual** | **Seguridad Social** | **Sueldo anual + SS** | **Coste por hora** |
|---|---|---|---|---|
| Programador (P) | 17.800€ | 5340€ | 23.140€ | 10€/h |
| Tester (T) | 17.800€ | 5340€ | 23.140€ | 10€/h |
| Gestor proyecto (G) | 35.600€ | 10.680€ | 46.280€ | 20€/h |
| Escritor técnico (E) | 35.600€ | 10.680€ | 46.280€ | 20€/h |
| Enc. Búsqueda de Información (I) | 35.600€ | 10.680€ | 46.280€ | 20€/h |

Tabla 2: Sueldo del personal anual y por horas

El sueldo anual se estima por las horas de trabajo anuales que son 1.780 según asesorias.com, por lo que el sueldo anual se calculará multiplicando el precio por hora de los roles y las horas anuales.

### Costes por actividades (CPA)

A continuación en la tabla 3 se recoge los costes por actividad del proyecto.

| Actividad | Importe (€) | Rol | Horas |
|---|---|---|---|
| Introducción, contexto, y alcance del proyecto | 200+200 | E,G | 10 |
| Planificación del proyecto | 200+200 | E,G | 10 |
| Presupuesto y Sostenibilidad | 300+300 | E,G | 15 |
| Documento Final | 300+300 | E,G | 15 |
| Búsqueda De Información sobre el empresas | 300 | I | 15 |
| Búsqueda Información (IBM) | 400 | I | 20 |
| Búsqueda Información (Microsoft) | 400 | I | 20 |
| Búsqueda Información (Oracle) | 400 | I | 20 |
| Organización de la información y Comparativa | 500+500 | I,E | 25 |
| Reuniones de seguimiento | 100 | G | 10 |
| Planificación de la aplicación | 200 | P | 20 |
| Familiarizarme con el entorno y lenguajes de programación | 300 | P | 30 |
| Desarrollo Aplicación | 800 | P | 80 |
| Reunion de seguimiento | 100 | G | 10 |
| Testear Aplicación y errores | 250 | T | 25 |
| Integrar aplicación a la Blockchain (DApp) | 150 | P | 15 |
| Documentación | 800 | E | 40 |
| Preparación de la presentación | 400 | G | 20 |
| Coste por actividad total | 7600€ + 0.3(7600€) | | |

Tabla 3: Coste del personal por tarea y horas

## Costes generales (CG)

### Costes directos

#### Software

Todo el software que usaré será gratuito.

Hardware

- ordenador portátil
- smartphone
- monitor
- mouse

Para calcular la amortización del hardware consideraré su uso diario durante los fines de semanas y días laborables, en total 130 días del proyecto. Con la extensión fueron 160 días, pero en los cálculos no se consideran al ser poco significativos 75h en 3 meses.

Se tiene en cuenta que el portátil tiene 3 años de uso y se refleja a la hora de calcular la amortización.

Amortización = (Precio_Inicial * dias_uso) / (dias_uso_anual * 4 años)

Se consideran dias_uso_anual como 320 días y con un tiempo de vida útil de 4 años.
Por lo que se amortizara el portatil usado:
1000€ * (3*320 + 130) / (320*4) = 851.56€ si se calcula desde el momento que se adquirió el portátil y 101.56€ si no se tienen en cuenta los 3 años de vida anteriores.

Amortización del segundo ordenador:
Depende del uso que se haga del portátil, pero si se realiza el cálculo anterior.
1000€ * (130 días) /320*4 = 101.56€

El resto del hardware no tiene un uso tan intenso como el portátil y su sustitución es viable con fondo para contingencias.

## Costes indirectos

Electricidad

Según tarifadeluz.com la luz general es de 0.104€/kwh. De este modo si nuestro proyecto dura 410h tenemos que el coste de luz será :

200W/h consumo del portátil * 410h = 82000W = 82KW/h -> 8.53€

30W/h consumo monitor * 410h = 12300W = 12.3Kw/h -> 1.3€

Total consumo (€) = 9.83€

## Acceso a internet

El precio del internet cuesta 40 euros/mes, dado que la duración del proyecto es de 4 meses y de media unas 4 horas de trabajo diarias, tenemos que el coste total de acceso a internet es de:

4 meses * 40€/mes * 4h/24h = 26,6€.

| | Coste | Comentarios | Horas |
|---|---|---|---|
| Hardware | | | |
| Laptop | 101.56€ | Precio de compra: 1000€ | 410 |
| Smartphone | 25.39 | Precio de compra 250€ | 50 |
| Monitor | 8.1€ | Se conecta al laptop por HDMI<br>Precio de compra 80€ | 410 |
| Mouse | 2.03€ | Precio de compra 20€ | 410 |
| Software | | | |
| Git | 0.00€ | Gratis | |
| VSC | 0.00€ | Gratis | |
| Java | 0.00€ | Gratis | |
| Solidity | 0.00€ | Gratis | |
| IBM platform | 0.00€ | Gratis | |
| Google Drive | 0.00€ | Gratis | |
| Gmail | 0.00€ | Gratis | |
| Truffle | 0.00€ | Gratis | |
| Costes Indirectos | | | |
| Electricidad | 9.83€ | | 410 |
| Internet | 26,60€ | | 410 |
| Espacio | 0.00€ | En lugar de trabajo será mi casa | 410 |
| Total CG | 173,51€ | | |
| Coste Total (Total CPA + Total CG) = 9880 + 173.51 = 10053,51€ | | | |
| Contingencia | 15 % de 10053,51€ = 1508.02€ | | |
| Riesgo: 1000 euros para un ordenador nuevo en caso que el usado se estropee, 101.56€ de | | | |

| amortización |
| --- |
| Total (CPA + CG + Contingencia + Riesgo) = 12663.09€ |

Tabla 4. Cálculo de costes generales

## 5.2. Control de gestión

Como con todo presupuesto pueden haber contratiempos que alteren las estimaciones hechas, por esa razón lo mejor es tener un control lo más estricto posible para comprobar si se está siguiendo lo planificado o el presupuesto se está gastando demasiado rápido o si está sobrando dinero. Para este fin es útil calcular posibles desviaciones una vez finalicen las tareas, como por ejemplo en el número de horas de trabajo empleado por parte de los miembros del proyecto o las amortizaciones de las herramientas de desarrollo.

Desviación de recursos humanos: Puede suceder que se esté empleando más horas de las estimadas o bien menos.

$$D.R.H = \sum_{i \in personal} (coste\_estimado\_hora(i) - real\_coste\_hora(i)) \times total\_horas\_reales(i)$$

Desviación de amoritzaciones: En caso que usemos el hardware más de lo planificado o menos puede variar el coste de amoritzaciones.

$$D.A = \sum_{i \in horas} (Horas\_uso\_Estimado(i) - Horas\_uso\_Real(i)) \times precio\_hora(i)$$

Desviación total

$$Desviación\ coste\ total = coste\_general\_estimado - coste\_general\_real$$

Con ayuda de estos cálculos podemos identificar cual es la tarea que está causando una desviación del presupuesto y valorar si es necesario utilizar las reservas para contingencias.

# 6. Sostenibilidad y compromiso social

| | Ambiental | Económico | Social |
|---|---|---|---|
| PPP | Consumo de diseño | Coste de desarrollo | Beneficiarios |

Tabla 5. Matriz de sostenibilidad

## Ambiental

Hito inicial
Se ha realizado una estimación de las herramientas y recursos que se pueden necesitar durante el desarrollo del proyecto. Teniendo en cuenta las fases del proyecto y los requerimientos tecnológicos necesarios, así como el impacto medioambiental que puede derivar del uso de estos recursos.

Primeramente el ordenador portátil que usaré será un ASUS g551j y se utilizará durante todo el desarrollo.
El ordenador será un ordenador portátil personal de 3 años de antigüedad y se optará por utilizarlo durante el desarrollo, aunque existe el riesgo de que durante el proyecto puedan haber problemas con el rendimiento o incluso que se llegue a estropear. En ese caso existe un riesgo que se tendrá en cuenta a la hora de estimar los costes del proyecto ya que tendré que hacer un gasto extra en adquirir otro portátil de características similares. El precio del portátil fue de 1000€.

En cuanto a la energía eléctrica que se necesitará dependerá del uso que se haga del portatil y el monitor, segun DisplayEspecifications.com el consumo medio de un monitor está en 30W/h y según ganaenergia.com está entre 150-200W/h, como el portátil es gaming requiere bastante energía por lo que se considerará que gasta 200W.

La parte que mayor efecto negativo puede tener en el medio ambiente es el despliegue y puesta en marcha de la aplicación en el ecosistema blockchain ya que si va dedicado a producción y tiene que estar disponible para múltiples usuarios, se deberá contratar un servidor que deberá permanecer encendido todo el tiempo.
El estar usando un servicio de otra empresa hace que no podamos medir el consumo de energía de nuestro proyecto en esa etapa. Pero un estudio llevado a cabo por Microsoft pone en manifiesto como el cloud computing tiene la capacidad

para que la ejecución software sea más eficiente que una infraestructura propia y de esta manera reducir el consumo energético pero esto ya depende de la compañía.

En nuestro caso el desarrollo y despliegue será local por lo que no tendremos que contratar un servicio de hosting.

## Económico

Hito inicial

La estimación económica del proyecto  por actividades se ha llevado a cabo detalladamente en el punto anterior. También se ha considerado los sueldos por horas de los participantes buscando por internet y se han multiplicado por el número de horas que se invierten en cada actividad del proyecto y por último se ha sumado el porcentaje de la seguridad social. Y para controlar la sostenibilidad económica se ha calculado las posibles desviaciones del presupuesto una vez finalice cada tarea.
También se ha estimado otros costes como pueden ser contingencias y riesgos, además de cómo puede afectar al presupuesto final del proyecto.

## Social

Hito inicial

Personalmente este proyecto ayudará a adquirir experiencia en la búsqueda de información, compararla y sacar conclusiones. Además de aprender a cómo elaborar un proyecto adecuadamente, estructurando el tiempo de las tareas así como las fases de planificación necesarias como la documentación, impacto económico , de impacto ambiental, impacto social, etc.
El objetivo de la comparativa del servicio que ofrecen diferentes empresas es ahorrar tiempo a los interesados que quieran desarrollar una aplicación en un ecosistema blockchain y ayudarlos a identificar cual es el mejor framework que ofrece el servicio más adecuado. Posteriormente mediante el desarrollo de una aplicación con smart contract y su despliegue en una de las opciones que estudiadas se planea validar la información con un ejemplo funcional.

En cuanto a la aplicación, está enfocada a ayudar a grupos de trabajo  ya que su funcionamiento tiene que ver con la gestión de fechas de entrega límite (deadlines) de trabajo de tal manera que no haya un solo responsable durante el transcurso del proyecto y todos sean conscientes del ritmo de trabajo del mismo.
Ademaś la idea es poner en manifiesto la capacidad de los smart contracts con respecto a la automatización de tareas y de cómo puedan ayudar en distintos ámbitos y sustituir ciertos roles de suma importancia.

# 7. Comparativa

## 7.1. Motivación

La motivación de realizar una comparativa viene primeramente del interés por el blockchain y los smart contract que permiten automatizar tareas o roles de manera fiable. Cuando ciertas condiciones se dan y se cumplen las condiciones que el contrato define se pasa a ejecutar ciertas acciones programadas dentro del contrato. Además existe la posibilidad de aplicar esta tecnología a distintos ámbitos como el financiero, comercial, gubernamental, en salud, etc. Proporcionando así una manera más fiable, más transparente para el cliente y más segura al basarse en P2P.
Como una forma de aprender más sobre esta tecnología que me pareció tan interesante como para basar mi trabajo de fin de grado en comparar dos de estos framework y sacar en claro qué diferencias existían entre ellas, de qué manera podía afectar esto a los usuarios o a la información que se almacena dentro de ellas y si en general existía una mejor que otra.
De este modo se pretende centrar el proyecto en las redes blockchain privadas y con permisos (permissioned) que se diferencian de la de bitcoin, por ejemplo, en la identificación para entrar en una red privada y el resto de participantes deben autorizar el acceso. De este modo todos los participantes de la red se conocen entre ellos y saben quién serán los responsables de almacenar su información y validar futuros cambios, además todos deben conocer el contrato inteligente, instalarlo en sus nodos/peers y ponerlo en funcionamiento.

Al comparar dos modelos de redes privadas con permisos se pretende indicar las diferencias que existen entre este tipo de tecnologías y si es posible definir cuál es mejor.

## 7.2. Red Blockchain permisionada

En las redes privadas los permisos de escritura están restringidos aunque puede ser que cualquiera pueda consultar el estado de la red y de la base de datos que gestione.
Un consorcio o red con permisos es una red privada pero gestionada no por una única organización sino que por un conjunto. Entre sus características está que el consenso se lleva a cabo por nodos preseleccionados y al haber un número limitado y preestablecido de nodos, el consenso es más rápido que uno de proof-of-work, por ejemplo. Además de la preselección también hay una verificación de los nodos que interactúan con la red y que aprueban las transacciones.

Una modalidad de red en la que para realizar transacciones se ha de estar autentificado, además de tener permiso del resto de participantes, esto se consigue mediante certificados o claves que identifican a cada una de las organizaciones de la red, pueden ser socios, colaboradores de una empresa que realizan un trabajo conjunto. Como en la cadena de suministros donde es imprescindible que haya confianza en quienes trabajan conjuntamente o al menos que sepas el estado del producto hasta que llegue a tus manos.

## 7.3. Presentación de dos redes permisionadas

Empezaré definiendo las dos redes privadas con permisos que compararé.

### 7.3.1. Hyperledger Fabric

HyperLedger es una plataforma de blockchain de código abierto lanzada por la fundación Linux en 2015.
Fabric es un framework modular, que usa la tecnología de libro de cuentas descentralizado (DLT) y fue diseñado por IBM para uso empresarial.
Por lo tanto la unión de estos frameworks dio lugar a Hyperledger Fabric que es una red blockchain privada que requiere permiso para acceder por ello habrá menos nodos que hará más rápida la red y la información que se transmite puede ser simplificada sin necesidad de tanta seguridad en el transporte.

### 7.3.2. Ethereum Quorum

Etherum es una plataforma que adopta la tecnología blockchain creada por el programador Vitalik Buterin, es *permissionless* y pública que está orientado al despliegue de aplicaciones distribuidas y smart contracts con los que gestionar transacciones y automatizar resultados o tareas.

Quorum es una plataforma de blockchain orientada al sector empresarial. Es un fork del cliente de ethereum ‘geth’ con protocolos adicionales que satisfacen las necesidades empresariales.

## 7.4. Definición y estructura de un Ledger descentralizado

### 7.4.1. El blockchain y el world state

El *blockchain* o cadena de bloques es el conjunto de logs de las transacciones realizadas, que pueden ser consultas o modificaciones en el *world state*. Es decir, actúa como un registro histórico del ledger/libro de cuentas pero descentralizado.
Las transacciones se recogen en bloques que son enlazadas a la cadena de bloques, facilitando el entendimiento de los cambios que se han llevado a cabo y permitiendo que se haga un rastreo de estos cambios y llegar a ver los cambios pasados. Estas cadenas de bloques es inmutable y por tanto no se puede alterar de ninguna forma, para asegurar esto existen los header de cada bloque donde se guardan un hash de las transacciones contenidas en el bloque y además una copia del hash del bloque anterior, por lo que toda la cadena está enlazada criptográficamente, permitiendo así que si uno de las copias del blockchain se altera, los demás participantes de la red se podrán dar cuenta.
Su implementación es un fichero no una base de datos como el *world state.*
El primer bloque B0 (en la figura 5) contiene la transacción de configuración (génesis) que indicará el estado inicial de una red.

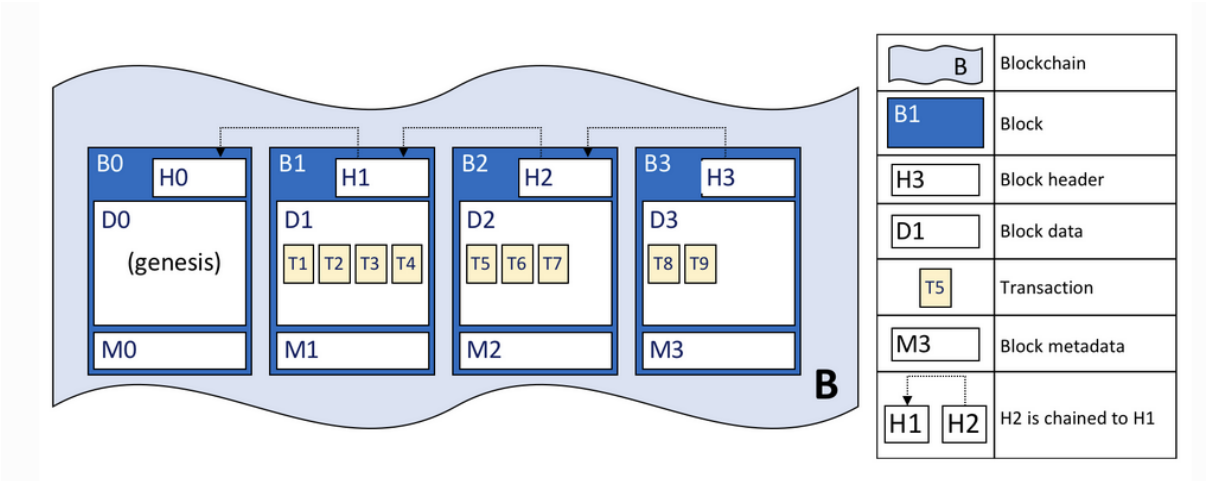


Figura. 5 : Representación cadena de bloques

El *world state* es una base de datos que contiene un caché de los valores actuales de un conjunto de estados del libro mayor/ledger. El *world state* facilita que un programa acceda directamente al valor actual de un estado en lugar de tener que calcularlo recorriendo todo el registro de transacciones. Los estados del libro mayor se expresan, de forma predeterminada, como pares clave-valor. El *world state* puede cambiar con frecuencia, ya que los estados se pueden crear, actualizar y eliminar.

## 7.5. Componentes de la red

### 7.5.1. Hyperledger Fabric

Peers / nodos: Cada organización de la red deberá tener al menos un peer que hará de participante en la red.

Peers / nodos de ordenamiento: La red deberá tener al menos un peer de ordenamiento que servirá para gestionar la red, canales y el consenso.

Autoridad de certificación: Cada organización debe tener su propia autoridad de certificados, así como el servicio de ordenamiento. Los certificados permiten que los peers participen en la red.

Dapp: Diferente tipo de aplicaciones, puede ser nativa, web, portatil (android) y se conecta con la red mediante SDK / API y con los requisitos de autentificación.

### 7.5.2. Ethereum Quorum

Quorum node: Es un nodo participante en la red públic/privada, tendrá un identificador que servirá para agrupar nodos en redes privadas. Tendrá una base de datos fragmentada ya que la información de las transacciones privadas no debe ser accesible por cualquiera.

Tessera node: Es un gestor de privacidad, una instancia de constellation que se utiliza para proporcionar privacidad y seguridad a las transacciones de la red.

Enclave: Se encarga de encriptar / desencriptar datos.

Transaction manager: Es un participante de la red Constellation, el cual se comunica con otros participantes (otros transaction manager) que se comunica con otros Quorum node y permite el intercambio de la información de manera segura.

Dapp: Diferente tipo de aplicaciones, puede ser nativa, web, portatil (android) y se conecta con la red mediante SDK / API y con los requisitos de autentificación.

## 7.6. Cómo consiguen la privacidad con permisos

### 7.6.1. Hyperledger Fabric

Distintas organizaciones tienen permiso mediante identificación, a través de certificados digitales que proporciona una CA, para poder interaccionar con la red de lo contrario no podrán ni ver la información de las transacciones ni tampoco podrán procesar los bloques que se registran en la cadena de bloques además de iniciar peticiones a la red. De esta manera se adquiere privacidad con las características de transparencia, inmutabilidad y seguridad que ya da blockchain.

Las CA son las *certificates authority* que son entidades de plena confianza en la red, de esta manera entregan los certificados que identifican a cada nodo de la red de manera absoluta y no hay duda por parte de ningún integrante de la red que todos son de fiar.
En el caso de Hyperledger Fabric existen tres tipos de CA:

enrollment certificate authority (**ECA**) o autoridad de certificados de inscripción: Proporciona certificados a los nuevos usuarios que quieran registrarse en una red blockchain y permite solicitar un par certificado de inscripción que uno será para firmar los datos y el otro será para encriptar los datos, las claves públicas incrustadas deben ser del tipo ECDSA.

*transaction certificate authority* (**TCA**) o autoridad de certificados de transacción:
Proporciona certificados la capacidad para desplegar chaincode y también para invocar transacciones de chaincode a la red blockchain.

*TLS certificate authority* (**TLSCA**): Proporciona la capacidad para asegurar las comunicaciones en su canal.

Además existe una jerarquía según el servidor donde estén instalados los CA.
Primero existe un Root CA que da confianza a los certificados y luego en menor nivel están los intermediate CA que certifican a los participantes (peer, ordenamiento, administradores, clientes).

Membership service provider (**MSP**) o membresía de servicio de proveedor: Abstrae todos los mecanismos y protocolos criptográficos detrás de la emisión de certificados, la validación de certificados y la autenticación de usuarios. Una red blockchain de Hyperledger Fabric puede estar gobernada por más de una MSP y un MSP identifica a todos los participantes de la misma. Esto proporciona modularidad de las operaciones de membresía e interoperabilidad entre diferentes estándares y arquitecturas de membresía.

Una identidad MSP es válida si y únicamente si existe un único camino de root CA al certificado digital expedido.

Se identifica un MSP con un MSP ID sirve para identificar un consorcio, una organización o una división de una organización y para poder referenciarla en la red. Consta de listas de certificados que identifican el rol, además también se permite añadir *certificate revoke list* (CRL) es para evitar dar acceso a ciertos certificados o que son expedidos por ciertos CA que no son de confianza, además también sirven para prohibir el acceso a las entidades que ya no forman parte del consorcio.

Además existe una estructura lógica llamada **canales** que conecta la/las dapp con los nodos de la red de manera que se pueden formar diferentes grupos de organizaciones en los que está asegurada la privacidad entre ellos y sumados a los certificados y MSP se consigue la privacidad.
Estos canales proporcionan una partición de los datos mediante la creación de una subred privada de comunicación hacia dos o más miembros de la red que podrán ejecutar transacciones en dicho canal y que serán visibles únicamente por aquellos participantes que se hayan autenticado tengan permiso para entrar. Se puede pertenecer a diferentes canales si se tienen los permisos necesarios y de esta manera intercambiar información confidencial con diferentes grupos de trabajo dentro de una misma red. La autentificación de un nodo se la da el MSP y de esta manera se le identifica a cada canal y servicios.

Un elemento importante en el papel de los canales es el nodo de anclaje (anchor).
**Anchor peer**: El rol de este nodo es permanecer visible por todos los nodos de la red. Esto sirve para poder comunicar con los nodos de otras organizaciones, si una organización no tiene este rol, un nodo de la organización solamente podrá descubrir nodos de su propia organización participando en un mismo canal. Es decir, si una organización no tiene el nodo de anclaje cuando un participante se una al canal provocará que la organización no tenga conocimiento de este nuevo participante. Un nodo de anclaje puede pertenecer a diferentes canales pero esto no hará que las transacciones de diferentes canales se puedan compartir entre ellas, por lo que esto hace que una empresa pueda coexistir con competidores de su negocio en una misma red.

### 7.6.2. Ethereum Quorum

Networking and cryptography library (NaCl): Es una librería de software de alta velocidad para comunicación de red, criptografía, desencriptado, firmas digitales.

Ethereum Quorum utiliza un gestor de privacidad (off-chain mechanism) que está compuesto por un gestor de transacciones (**Tessera**) y un enclave (ver figura 6).
Tessera permite la distribución de transacciones privadas en Quorum. Cada nodo que tenga instalado Tessera deberá:

- Generar y mantener un número de parejas de claves privadas/públicas
- Autogestiona y descubre otros nodos mediante su clave pública, únicamente hace falta conectarse a un nodo que esté conectado a la red privada.
- Proporciona una API privada y pública para comunicarse con otros gestores de transacciones. La privada sirve para comunicarse con los nodos Quorum y los públicos sirven para conectarse a otros nodos que actúan como gestor de transacciones.
- Permite una lista blanca por IP.
- Utiliza certificados TLS y otros modelos de confianza como Trust On First Use (TOFU), autoridades de certificados, listas blancas.

En una red de permisos es necesario activar el flag de permisos que consultará el fichero **permissioned-nodes.json** donde están los nodos que serán parte de la red y de los cuales y hacia los que se podrán aceptar conexiones.

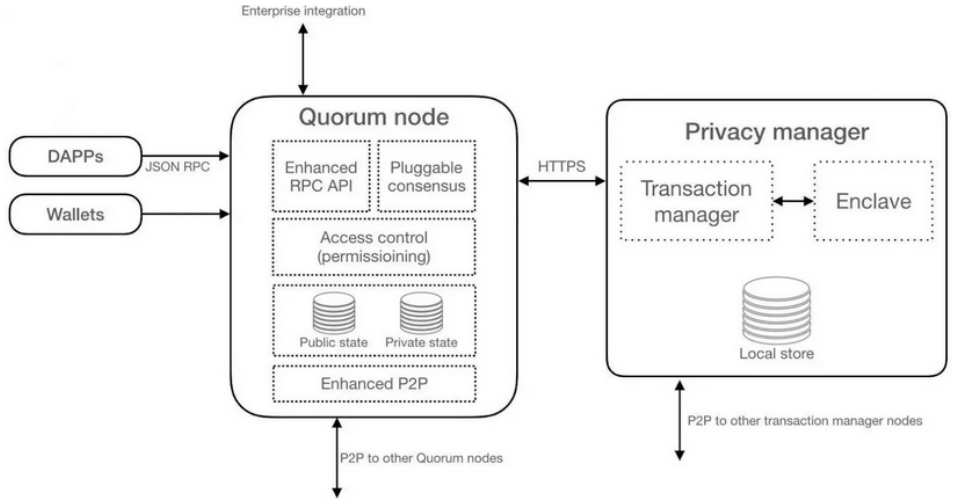


Figura 6. Arquitectura Ethereum Quorum

## 7.7. Seguridad de la información

### 7.7.1. Hyperledger Fabric

Hyperledger Fabric permite encriptar los datos antes de llamar al chaincode pero además deberá proporcionarse medios para compartir los datos fuente y también una forma de compartir las claves de desencriptado.
Una última manera de asegurar los datos es a partir de un sistema de ficheros de encriptación en el nodo y la información en tránsito es encriptada vía TLS. Esto significa que la información será encriptada a nivel de sistema y trabaja sobre NTFS. Esto se suma a las medidas de seguridad de la red como los canales y MSP.

### 7.7.2. Ethereum Quorum

Quorum **enclave** es otro componente del gestor de privacidad y se encarga de gestionar la encriptación y desencriptación de los datos privados y de la gestión de las claves de identificación de los usuarios. Además también entra la segmentación que se aplica a cada base de datos local de nodos que únicamente es accesible por el mismo nodo. De esta manera la información solamente podrá ser consultada o modificada por aquellos miembros que pertenezcan al grupo de transacciones privadas que habrá creado y tengan acceso al contrato privado. Existen claves públicas que identifican a los grupos y por lo tanto harán las transacciones privadas para aquellos grupos.

## 7.8. Consenso de la red

El consenso es el proceso por el cual los nodos aceptan una transacción y determina el orden en el que deberá ser guardada en un bloque que se unirá criptográficamente a la cadena de bloques.

### 7.8.1. Hyperledger Fabric

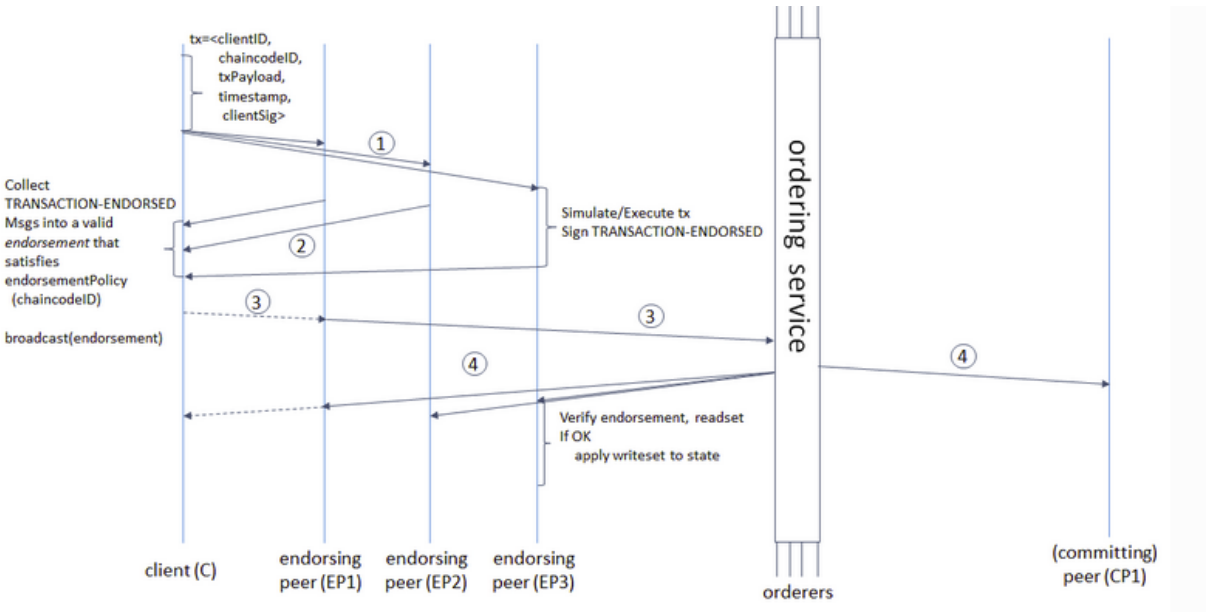


Figura 7. Esquema de la función del ordering service

Hyperledger Fabric utiliza el protocolo de ordenamiento que es un algoritmo determinístico que empaqueta en bloques las transacciones aprobadas y las envía a los nodos a través de los canales correspondientes mediante los nodos de ordenamiento. Los nodos de ordenamiento reciben transacciones concurrentes de diferentes aplicaciones y deben trabajar conjuntamente para agruparlas en una secuencia bien definida y almacenarlas en bloques que formarán parte de la cadena de bloques. La secuencia de transacciones dentro de un bloque no es necesariamente la misma en la que el servicio de ordenamiento las ha recibido de la aplicación ya que lo que es importante es que este servicio las pone en un orden estricto y ese orden es el que **todos** los nodos tendrán al validar y guardar las transacciones (ver figura 7).
Este nodo de ordenamiento ejecuta el binario Hyperledger Fabric de ordenamiento instalado en el nodo y únicamente realizará el orden cuando una invocación del chaincode se realiza. Por lo tanto, es la programación del chaincode la que dispara la ejecución del servicio de ordenamiento.

Existen tres mecanismos de consenso:

- Solo: No proporciona ni high availability (HA) ni descentralización.
- Kafka: Proporciona HA pero no descentralización.
- Raft: Proporciona ambas.

*RAFT-based Consensus*: Se basa en el modelo de Crash fault tolerant (CFT) consensus para una generación de bloques mayor, finalización de transacciones y creación de bloques a demanda. Su funcionamiento consiste en un líder y sus seguidores, en la red se elige un líder y su decisión es replicada por el resto. Para elegir el líder se utiliza la votación de la mayoría de los nodos disponibles.

*Istanbul BFT Consensus*: It is a Byzantine Fault Tolerant (BFT) algorithm which is based on Practical byzantine fault tolerant (PBFT)

La diferencia entre CFT y BFT es que CFT puede tolerar hasta un N/2 fallos en el sistema y BFT N/3 fallos. Pero CFT no puede garantizar que el consenso al que se llegue sea el correcto y BFT si.

### 7.8.2. Ethereum Quorum

Quorum al ser un fork de Ethereum que es una red pública tiene un característica heredada de que la red puede ser pública o privada. Por lo que la base de datos debe estar dividida en dos. Con la segmentación se consigue la separación y cada nodo tendrá una copia de su red privada y una común que será la pública. De esta manera el protocolo que se emplea para llegar al consenso de la red privada es QuorumChain.

**QuorumChain** es un protocolo de voto basado en la mayoría (raft o istanbul bft), donde un subconjunto de los nodos en la red tienen el permiso de votar los bloques y otros tienen el permiso de crear bloques asignándoles el **Block maker's role** con la correspondiente firma que los identifica. Para asignar los derechos de voto el Quorumchain está implementado dentro del smart contract, que además gestiona el proceso de consenso ya que sigue el rastro de la lista de votantes y makers.

- Global Transaction Hash: es un hash de todas las transacciones dentro de un bloque. Que representa el contenido de un bloque para reconocer si el contenido ha sido modificado.
- Public State root hash: Este hash permite comprobar si un nodo tiene la misma cadena de bloques dentro de sus base de datos. Y de esta manera exista sincronización con el resto de nodos de la red privada.

Quorum proporciona una llamada a la API, **eth_storageRoot**, que devuelve el hash de estado privado para una transacción determinada a una altura de bloque determinada, que opcionalmente se puede llamar en la capa de aplicación para realizar específicamente una validación de estado fuera de la cadena con una contraparte.

## 7.9. Smart contract y despliegue

Un contrato inteligente es el código, denominado a veces código de encadenamiento, con el que interaccionan las aplicaciones para leer y actualizar datos en el libro mayor de blockchain.

Para ponerlos en funcionamiento primero se necesitan instalar en uno de los nodos y posteriormente instanciarlo en el canal.
Todo esto se puede llevar a cabo mediante una API o SDK.

### 7.9.1. Hyperledger Fabric

Chaincode: Es un conjunto de smart contract.
¿Qué características debe tener?

- Desarrollado con Go, Node.js, Java o Javascript.
- Acordado y verificado por todos los miembros de una red blockchain
- Política de aprobación: La política de aprobación especifica el conjunto de organizaciones en un canal que puede ejecutar el contrato inteligente y **validar,** mediante firma, las transacción.
- recopilación de datos privados: se utilizan para mantener la información confidencial privada de los miembros de otras organizaciones **en un canal**.
    - gossip nodo

Los contratos inteligentes se instalan en los nodos mediante una API o SDK y para que pueda ser accesible a otros nodos primeramente se deberá instanciar en el canal. Una vez esté instanciado, el resto de nodos podrán instalarlo en su memoria física.
Un contrato inteligente se define por su **nombre y versión** por lo que debe ser coherente en todos los nodos. Una vez instalado en los nodos uno de ellos inicia una instancia en el canal que configura el libro de cuentas con la información contenida en el contrato inteligente.

### 7.9.2. Ethereum Quorum

¿Qué características tiene?

- Desarrollado con Solidity.
- Puede ser privado / público.
- un contrato privado no se puede cambiar a público y viceversa.
- Existe un campo en el smart contract que indica la visibilidad del mismo para otros nodos de la red, el privateFor que es una lista de publicKey (identificadores de los nodos).

IBM, Oracle y Microsoft permiten desarrollar smart contract con distintos lenguajes de programación, como javascript, Go, solidity. Además aporta una extensión para desarrollarlos a través de Visual Studio Code, que es una herramienta que agiliza y facilita el desarrollo, el despliegue y la depuración del código.

## 7.10. Flujo de las transacciones

El flujo de las transacciones (ver figura 8) depende si el ledger (libro de cuentas digital/base de datos) cambia. Los dos apartados detallan dos de las transacciones más utilizadas que son la de modificación y de consulta del ledger.

### 7.10.1. Hyperledger Fabric

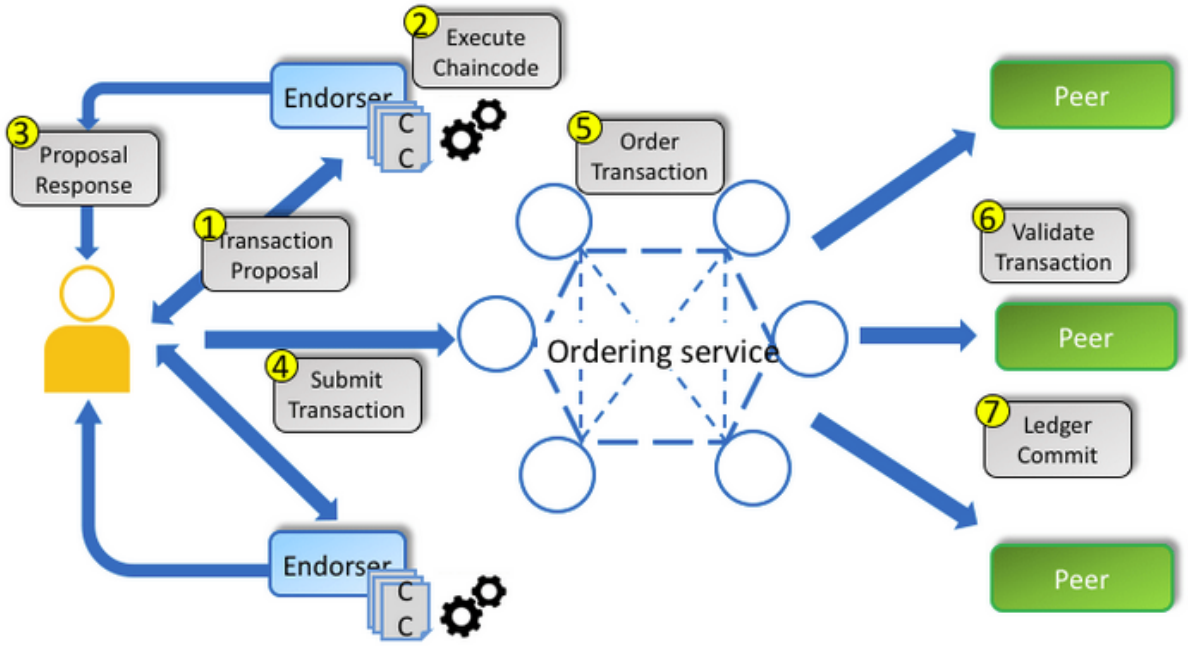


Figura 8. Flujo de una transacción de actualización de Hyperledger Fabric

- Txr. Modificación: Desde la aplicación distribuida se conecta al canal de la red privada, si está autorizado, mediante una API o SDK. Se realiza una propuesta de transacción a los nodos de aprobación, estos ejecutan el chain code/smart contracts y se comprueba si la petición es válida para el usuario (si está autorizado) y se envía una respuesta a la propuesta. Si tiene permiso se realiza la transacción y se envía la transacción a un nodo de ordenamiento que tendrá que trabajar conjuntamente con el resto (servicio de ordenamiento) para llegar a un consenso en el orden de las transacción y guardarlo en un bloque. Una vez hecho se envia el bloque a los peer (nodos donde se guardan las cadenas de bloques) y estos validan que la información sea correcta a partir de:
    - Firma de los entidades
    - Firma de los nodos de aprobación
    - Verificar que la política de aprobación del chaincode

Una vez se comprueba que todo es correcto se pasa a guardar en el ledger el bloque en cuestión y se notifica al cliente de la dapp.

- Txr. Consulta: Desde la aplicación distribuida se conecta al canal de la red y si se tiene permiso para realizar la transacción se realiza la consulta al ledger de un peer de la red privada y se devuelve la información al cliente de la dapp.

### 7.10.2. Ethereum Quorum

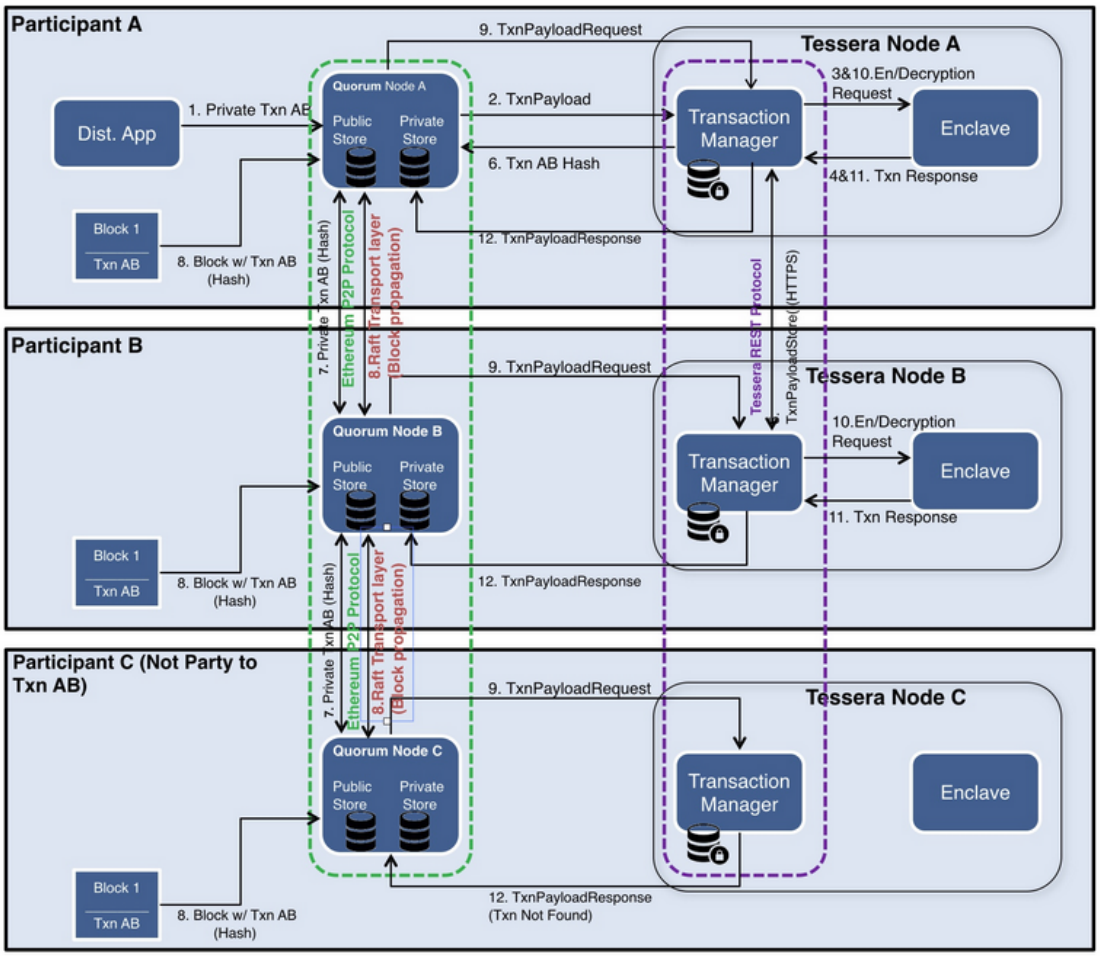


Figura 9. Flujo de una transacción de actualización de Ethereum Quorum

- Trx. modificación: Desde la aplicación distribuida se conecta a la red mediante una API o SDK que llega a un nodo Quorum donde está instalado el smart contract y si desde la aplicación se proporciona la clave pública privada necesaria se iniciará la transacción. La transacción es igual que una transacción en Ethereum, esto se debe a que Quorum es un fork de Ethereum y comparte muchas características públicas. La transacción para que sea privada se manda a un nodo Tessera (gestor de privacidad) mediante una API. Dentro de este nodo se procede a encriptar los datos con ayuda del enclave, este responde al gestor de transacciones que manda los

datos encriptados a otros nodos Tessera que pertenezcan al grupo privado y envía al nodo Quorum un hash que representa los datos de la transacción. El nodo Quorum por lo tanto envía mediante el protocolo de Ethereum p2p (Ethereum wire protocol) el hash a todos los demás nodos de la red.
Una vez tenga el hash se pasa a pedir los datos al nodo Tessera, éste lo desencripta con ayuda del enclave y lo envía a los nodos Quorum que se pertenezcan al grupo privado y se puedan identificar con su clave pública, de esta manera no se enviará los datos a los nodos públicos o de otro grupo privado, dando error en la transacción. Los nodos Quorum llegaran a un consenso con el orden de las transacciones (datos/payload) y los que tengan permiso para crear bloques con los datos desencriptados iniciaran el proceso de propagación de bloques con el protocolo de Ethereum p2p que se guardaran en la parte privada de la memoria de los nodos Quorum.

- En el caso de las consultas únicamente haŕa falta que la aplicación proporcione los permisos al nodo Quorum, mediante el SDK. Una vez dentro se podrá acceder a la base de datos pública y a la privada si se identifica con la clave criptográfica correspondiente.

## 7.11. Despliegue de la red

Estos framework se pueden desplegar en una máquina local con el código fuente correspondiente. Pero también es posible desplegar una red desde la plataforma que la ofrece como servicio de tal manera que se ahorra el tener que configurar componentes, descargar dependencias, librerías etc. A continuación unos ejemplos.

### 7.11.1. Hyperledger Fabric en IBM y Oracle

Como se ha indicado Hyperledger Fabric es modular, esto quiere decir que el sistema está dividido en módulos donde cada red puede ser independientemente creada, modificada, reemplazada.
Para este objetivo los contenedores son la solución que implementa IBM por ello habrá que utilizar un servicio de Kubernetes para desplegar una red blockchain.

Será necesario crear una instancia de Kubernetes y los elementos mínimos necesarios serán un par de nodos de organización, un nodo de ordenamiento, tres entidades emisoras de CA, un servicio de ordenamiento, dos peers (las base de datos couchdB en IBM) y un canal.

### 7.11.2. Ethereum Quorum en Azure

En el caso de Quorum con portal Azure se puede desplegar la red y gestionar el número de nodos que se va a utilizar. En el caso de Azure existen una serie de

servicios necesarios para conectarse y realizar transacciones con la red Blockchain, como Logic App y Service Bus.

También se puede desplegar con la configuración de scripts y el código de Quorum y todas sus dependencias que se encuentran en Github como software libre.

## 7.12. Rendimiento

La tecnología distribuida es popular por su gran disponibilidad al distribuir la información en diferentes nodos, pero según el objetivo que tenga la red privada no solo la disponibilidad es importante sino también la latencia, y ancho de banda que pueda soportar. De esta manera hay que ser conscientes que la red debe ser capaz de soportar cientos o miles de transacciones por segundo.

Si por ejemplo se orienta al sector financiero donde un banco debe procesar las transacciones que hagan los clientes en este caso se habla de miles de transacciones por segundo, ya que podrán realizarse o bien desde un cajero, desde casa con un ordenador o bien desde el móvil.

Si en cambio se enfoca más a una cadena de producción donde habrá un seguimiento de los productos desde materia prima hasta producto terminado, el número de transacciones por segundo no serán tan numerosas.

Para Hyperledger Fabric que está enfocado a la cadena de producción el número de transacciones por segundo son muy importantes y puede llegar hasta 3.000TPS, aunque existen investigadores que han conseguido que llegue hasta los 20.000TPS mediante optimizaciones.
Por otro lado Ethereum Quorum puede llegar hasta las 100 transacciones por segundo.

La **capacidad de los bloques** es otro factor que afecta al rendimiento, cada bloque guarda un conjunto de transacciones que puede variar según el bloque aunque nunca superará el máximo establecido en el contrato inteligente. Aumentar este máximo hará que aumente el rendimiento de las transacciones dentro del bloque, ya que puede ser el caso que un cierto número de transacciones que requieran dos bloques a procesar por los nodos pasen a requerir uno.
Las desventaja es que se necesitará más poder de computación para poder validar esos bloques más grandes, también se incrementarán los requisitos de memoria y el ancho de banda.

## 7.13. Casos de uso en empresas

### Introducción

La tecnología blockchain ha venido ganando popularidad y como prueba existen diversas empresas que ya utilizan esta tecnología para dar fiabilidad y transparencia a sus servicios. A continuación unos ejemplos para cada uno de los dos frameworks.

### 7.13.1. Hyperledger Fabric


Es una cadena de tiendas que vende comestibles al por menor y utiliza la tecnología blockchain de IBM para llevar a cabo el seguimiento de sus productos, desde que la granja hasta que llega a las tiendas.
La motivación se debe a las regulaciones que el gobierno de estados unidos les impone ya que se debe ser transparente con el producto que se está vendiendo y saber que tan saludable puede ser y evitar la mezcla con productos que no garanticen el nivel de calidad.


Se utiliza la tecnología blockchain para sustituir los votos delegados tradicionales sin tener que renunciar al derecho de anonimato y se añade la transparencia y eficiencia dentro de una corporación sobre la toma de decisiones.

True Tickets trabaja con Chateaux Software Development Inc., y IBM Blockchain partner, para construir un sistema de tickets en la plataforma blockchain de IBM.
La solución de etiquetas tiene dos resultados críticos:

- Crear y preservar la gobernanza del tiquet identificando a todos los compradores y vendedores y garantizando que son legítimos.
- La plataforma de tickets es inmutable, permite a los artistas, conciertos, promotores y fans de seguir el camino a través de cada ciclo de su vida desde su creación hasta el uso en el evento.

### 7.13.2. Ethereum Quorum

Desarrolló Krispay qué es la primera cartera de millas digitales basada en blockchain que permite a los clientes a pagar con sus millas viajadas instantáneamente en los comercios asociados con Singapur airlines, directamente desde sus móviles.
Se basa en millas aéreas a la hora de realizar transacciones a la red blockchain y guardar token en su cartera digital que luego podrán gastar en productos.

La compañía de café utiliza la tecnología blockchain para registrar el origen del café así como el camino que ha seguido hasta la llegada a una de sus tiendas. A partir de una etiqueta impresa en el paquete de café se puede utilizar la app y saber todo el camino que ha seguido desde la cosecha hasta tus manos.


Asociación Nacional de Corredores de Valores Automatizado de Cotización

Es la segunda bolsa de valores automatizada y electrónica más grande de los Estados Unidos, siendo la primera la Bolsa de Nueva York, con más de 3800 compañías y corporaciones. Tiene más volumen de intercambio por hora que cualquier otra bolsa de valores en el mundo. Más de 7000 acciones de pequeña y mediana capitalización cotizan en la NASDAQ. Se caracteriza por comprender las empresas de alta tecnología en electrónica, informática, telecomunicaciones, biotecnología, y muchas otras más. Dispone de un framework modular que ofrece una gran variedad de servicios y aplicaciones empresariales que combinado con el servicio blockchain de Microsoft, se reduce la complejidad ya que lo han adaptado para que contenga las características intrínsecas del libro de cuentas distribuido sin que se tengan conocimientos previos en la tecnología.

## 7.14. Cuadro comparativo

A continuación en la tabla 6 se recogen características de las redes para Hyperledger Fabric y Ethereum Quorum.

| Características | HyperLedger Fabric | Ethereum Quorum |
|---|---|---|
| Privacidad Datos y validación | Canales<br>Nodos de aprobación | Gestor de privacidad (Tessera)<br>- Gestor de transacciones<br>- Enclave |
| Transaction per second (TPS) | 3000 | 100 |
| Consenso | Servicio de ordenamiento | QuorumChain<br>Ethereum wire protocol (ETH) |
| Protocolo de consenso | Raft<br>Istanbul BFT | Raft<br>Istanbul BFT |
| Programación de smart contract | Go, Node.js, Java, Javascript | Solidity |
| Despliegue red | Kubernetes cluster de IBM | Software libre<br>Microsoft Azure Platform |
| Token<br>(Pago por validar bloques) | No requiere token | No requiere token |
| Identificación | Autoridades de certificado MSP (Membership service provider) | NaCl proporciona claves privadas y públicas que identifican a los nodos |
| Fork | - | Ethereum |
| Genesis block | Si | Si |
| Tipo de red | Privada | Privada/Pública |
| Encriptación | Opcional | Obligatoria |

Tabla 6. Resumen comparativo

## 7.15. Conclusión final de la comparativa

Según lo extraído de la comparativa está claro que Hyperledger Fabric es más eficiente ya que está construido especialmente para redes privadas, tiene más mecanismos de seguridad que Quorum, ya que ofrece diferentes puntos de comprobación a diferentes niveles, como a la hora de acceder al canal, los nodos de aprobación de transacción, la autentificación de bloques antes de añadirlos a la cadena, en el caso de asignar identificación a las organizaciones, los CA y MSP realizan un trabajo más completo que NaCl, ya que permiten asignar roles a los nodos y limitar sus acciones en la red de Hyperledger Fabric (HLF), esto ofrece mayor libertad a los usuarios a la hora de asignar permisos pero también ofrece mayor complejidad.

HLF no requiere encriptación de los datos y esto añade rendimiento a la red, porque reduce trabajo de computación, además el no requerir encriptación significa que la información no está expuesta a posibles intrusos, lo que no se puede decir de Quorum.

Además los canales llegan a proporcionar eficiencia al permitir que haya varios blockchain dentro de la misma red, así como ofrecer privacidad a los participantes de la red. En el caso de Quorum los gestores de privacidad realizan gran parte del trabajo en la red y esto puede actuar como cuello de botella.

# 8. Aplicación

## 8.1. Motivación

Para validar la información de la comparativa, se implementará y documentará una aplicación distribuida privada con permisos. Además de validar la información se expondrá la utilidad de los smart contract y de cómo hacen posible crear aplicaciones distribuidas y de cómo pueden estas aplicaciones ayudar a empresas, particulares y organizaciones de todo tipo a mantener una transparencia en sus transacciones, realizar acuerdos fiables, trabajar eficientemente, garantizar seguridad digital, etc. y qué beneficios puede dar a clientes de las aplicaciones.

## 8.2. Especificación de requisitos de la app

A continuación se listan las versiones con las que se trabaja en este proyecto.

node version >= 12

npm version >= 5

fabric network : 2.2.0

fabric-ca-client: 1.4.10

fabric-client: 2.2.0

## 8.3. Actores que participan en el funcionamiento de la app

Organizaciones: Serán los que firman digitalmente el contrato inteligente y los responsables de proporcionar los peers / nodos necesarios para el correcto funcionamiento de la red, además deberán llegar a un acuerdo entre ellas de las características de la red como el número de peers, el número de canales y su configuración, los peers de anclaje.

Autoridades de certificado: Son las entidades pertenecientes a cada organización y que se asegura que la identificación de todas las organizaciones participantes, asegurando la fiabilidad de la red y del sistema. Se basa en la expedición de certificados digitales que identifican a cada organización y los roles de sus integrantes.

Integrantes de las organizaciones: Serán los usuarios que utilicen la interfaz de usuario de la web app y que gestionan las tareas, es decir, el envío de tareas, la aprobación explícita de tareas, la consulta de tareas, la creación adicional de tareas que deberán aprobar el resto de participantes. Deberán registrarse para poder hacer el login.

## 8.4. Requisitos funcionales

A continuación se muestran una serie de tablas con los casos de uso de la aplicación.

### 8.4.1. Casos de usos

| | |
|---|---|
| Caso de uso 1 | Login de usuario |
| Actor principal | Integrante de una organización |
| Precondición | Certificado de usuario válido |
| Disparador | Escribir usuario, contraseña, email, teléfono y organización |
| Postcondición | Se crea una identificación |

Tabla 7. Caso de uso 1

| | |
|---|---|
| Caso de uso 2 | Registro de usuario |
| Actor principal | Integrante de una organización |
| Precondición | Identificación creada y firmada por un CA |
| Disparador | Escribir usuario, contraseña y organización |
| Postcondición | Redirección a la pantalla de progresos |

Tabla 8. Caso de uso 2

| | |
|---|---|
| Caso de uso 3 | Consultar listado de tareas |

| | |
|---|---|
| Actor principal | Integrante de una organización |
| Precondición | Usuario logueado |
| Disparador | Tabla con interfaz de filtrado |
| Postcondición | Listado de tareas con detalles |

Tabla 9. Caso de uso 3

| | |
|---|---|
| Caso de uso 4 | Enviar tarea |
| Actor principal | Integrante de una organización |
| Precondición | Tener tareas en progreso |
| Disparador | Botón |
| Postcondición | Tarea pasa a estado de waiting que indica que espera ser aprobada |

Tabla 10. Caso de uso 4

| | |
|---|---|
| Caso de uso 5 | Aprobar tarea explícitamente |
| Actor principal | Integrantes de las organizaciones |
| Precondición | Debe haber una tarea en estado de waiting y que requiera la aprobación de otro usuario |
| Disparador | Botón |
| Postcondición | La tarea se aprueba y pasa a estado finalizada |

Tabla 11. Caso de uso 5

| | |
|---|---|
| Caso de uso 6 | Consultar usuario |
| Actor principal | Integrante de una organización |
| Precondición | Existe usuarios en el Ledger |
| Disparador | Botón |
| Postcondición | Muestra la información de un usuario |

Tabla 12. Caso de uso 6

| | |
|---|---|
| Caso de uso 7 | Consultar listado de usuario |
| Actor principal | Integrante de una organización |
| Precondición | Existen usuarios en el ledger |
| Disparador | Buscar el nombre del usuario en el cajetín de filtrado |
| Postcondición | Muestra la información del usuario |

Tabla 13. Caso de uso 7

| | |
|---|---|
| Caso de uso 8 | Consultar listado de tarea específica |
| Actor principal | Integrante de una organización |
| Precondición | Existen tarea en el ledger |
| Disparador | Buscar el ID de la tarea en el cajetín de filtrado |
| Postcondición | Muestra la información de una tarea |

Tabla 14. Caso de uso 8

| | |
|---|---|
| Caso de uso 9 | Creación de tarea |
| Actor principal | Integrante de una organización |
| Precondición | Existen participantes en el ledger |
| Disparador | Especificar información para la tarea como la fecha de entrega y que participantes dependen de esa tarea |
| Postcondición | Creación de la tarea con la modificación del ledger y actualización de la barra de progresos |

Tabla 15. Caso de uso 9

## 8.5. Requisitos no funcionales

Los requisitos no funcionales son aquellos que son visibles por el usuario pero que no tienen ninguna relación directa con el comportamiento del sistema. De ese modo se puede identificar aspectos como el rendimiento, la apariencia / diseño de la interfaz de usuario, facilidad de uso, integridad y capacidad para registrar

**Apariencia / diseño de la interfaz de usuario**: Al ser una aplicación para demostrar el valor y utilidad de la plataforma blockchain, el diseño será simple y sencillo.

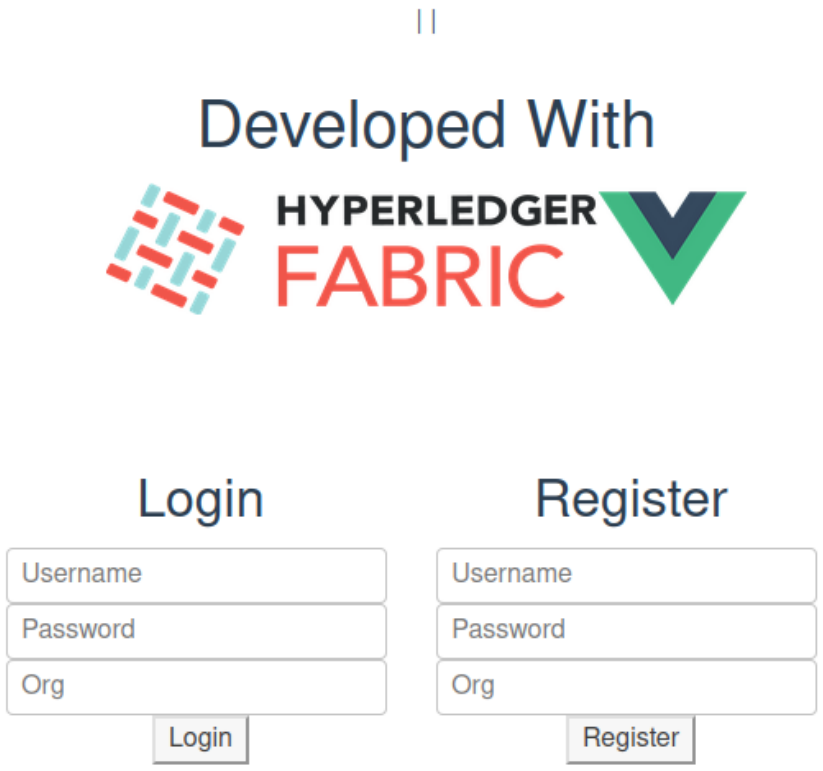


Figura 10. Portada de la aplicación web

En la figura 10 lo que se ve a primera vista es un formulario para realizar el Login del usuario con el Username y Password correspondiente, si no es correcto se verá un mensaje por pantalla conforme el usuario o contraseña no es correcto.
Luego se tiene otro formulario que sirve para registrar nuevos usuarios que formen parte de las organizaciones y tengan un certificado local de **miembros**, de no serlo no se podrá registrar en la aplicación.

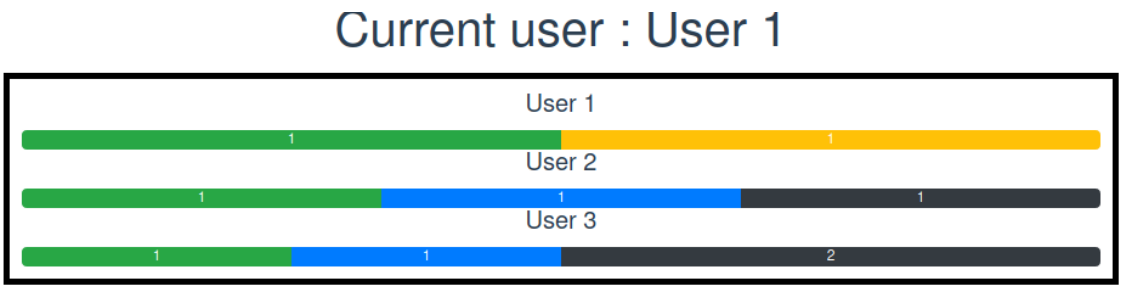


Figura 11. Barra de progresos que se visualiza al hacer log in

La figura 11 es lo que se obtiene una vez te identificas con éxito. Se observa barras de progreso de colores que representan:

Verde : Informa del número de tareas exitosas registradas en el ledger

Azul : Informa que la tarea actual no está lista todavía y el participante sigue trabajando en ella.

Amarillo : Informa que la tarea actual se esta procesando para pasar a estado finalizada y deberá ser aprobada explicitamente por el resto de participantes.

Negro: Informa del número de tareas que quedan por realizar.

Filter Type to Search Clear

| owner ID name | ID of the task | status of the task | Is Finished | Actions |
|---|---|---|---|---|
| User1 | 0 | finished | Yes | Show Details |
| User1 | 1 | waiting | No | Show Details |
| User2 | 2 | finished | Yes | Show Details |
| User2 | 3 | in_progress | No | Show Details |
| User2 | 4 | pending | No | Show Details |
| User3 | 5 | finished | Yes | Show Details |
| User3 | 6 | in_progress | No | Show Details |
| User3 | 7 | pending | No | Show Details |
| User3 | 8 | pending | No | Show Details |

Figura 12. Tabla de visualización de las tareas y detalles

La figura 12 muestra una tabla de todas las tareas con detalles.

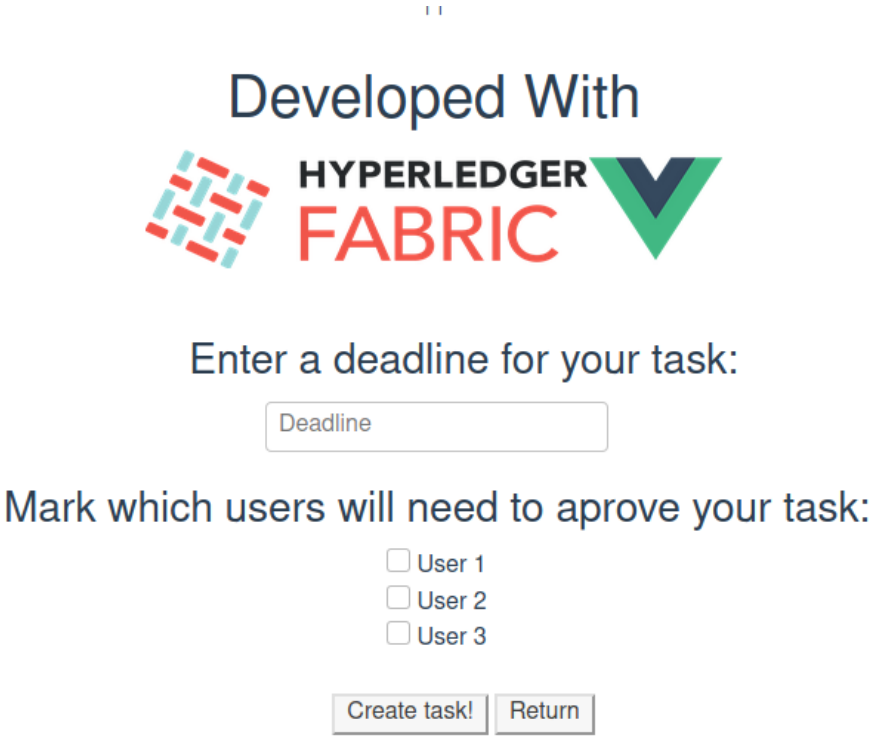


Figura 13. Pestaña de creación de tareas

**Facilidad de uso:** Como se ha comentado anteriormente esta aplicación se caracteriza por su sencillez ya que una vez logueado se puede observar el progreso de las tareas y realizar tus propias transacciones dándole a un botón. Y a la hora de registrar tareas también se señalan todos los campos necesarios a rellenar para crear una tarea exitosamente. Ver figura 13.

## 8.6. Arquitectura de la aplicación

### 8.6.1. Modelo 3 capas

La aplicación distribuida sigue el modelo 3 capas de presentación, lógica de negocio y datos.
La capa de presentación es la interfaz con la que el usuario interactúa que le muestra la información y partir de la cual se envía información al backend. En esta capa se realizan filtrados para evitar errores de formato, entre otros. Esta capa se comunica únicamente con la capa lógica.

La capa de lógica de negocios es la capa que recibe peticiones y se envían las respuestas tras el proceso. Es una capa donde definen una serie de reglas que han de cumplirse para acceder a los datos. En el caso de mi aplicación en un servidor

que atiende peticiones si el usuario se ha identificado correctamente y luego se procede a invocar el chaincode que hace querys sin ningún problema, pero la actualización de la información ya depende del consenso que se alcance en la red.

La capa de datos es donde residen los datos o activos y se reciben peticiones o envían datos a la capa de lógica de negocio. Al ser una aplicación distribuida todos los peers guardarán una copia de la información del ledger, dentro de Hyperledger Fabric existen dos modos bases de datos levelDB y CouchDB. La segunda ofrece realizar querys más complejas que levelDB.

### 8.6.2. Navegacion

A continuación en la figura 14 se puede observar la navegación entre los elementos de la web.

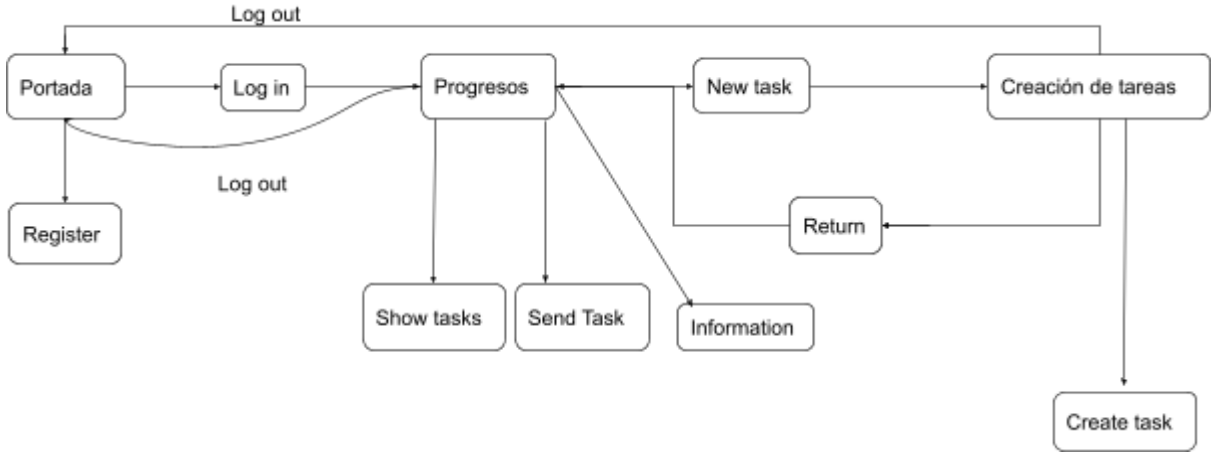


Figura 14. Diagrama de navegación

### 8.6.3. Componentes de la aplicación distribuida

En la figura 15 se puede observar un diagrama general de la aplicación.

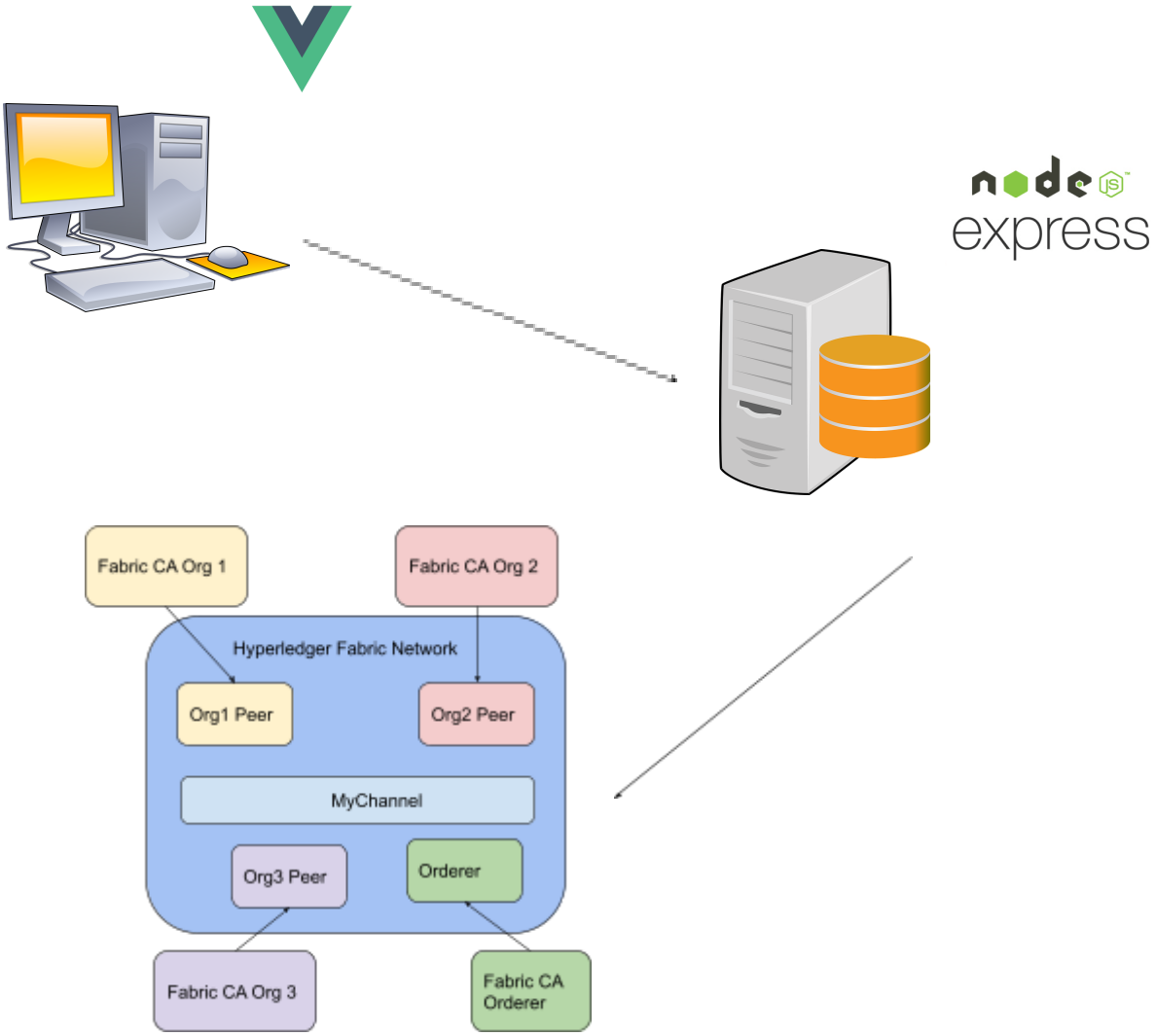


Figura 15. Diagrama de la aplicación distribuida [15] [16]

### 8.6.4. Lógica de la aplicación (Smart contract)

Las funciones básicas de la aplicación, se encuentran en el smart contract, y se encargan de crear, consultar, actualizar o eliminar (CRUD) los activos o assets del contrato. Esto quiere decir que sin acceso al contrato no se tiene acceso a los

activos y a este contrato solamente tienen acceso aquellas organizaciones autorizadas previamente.
Existen una serie de activos predefinidos en el contrato y son inicializados una vez se instala el contrato en la red. Al igual las funciones que están predefinidas y no se pueden añadir, quitar o modificar sin cambiar el contrato e instalar otro nuevo en la red. El contrato es inmutable.

El tipo de asset o activos serán tres. Las tareas, los usuarios y la información general. Ver figura 16.
Las **tareas** tendrán siete elementos:
La fecha límite de entrega: string
El ID de la tarea: int
El propietario de la tarea: string
El estado de la tarea: string
La organización a la que pertenece el propietario: string
Los usuarios que dependen de la tarea: array de string
Aprobación de los usuarios: array de bool

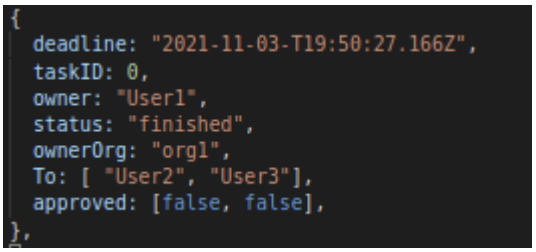


Figura 16. Definición de tarea

Los **usuarios** tendrán ocho elementos, ver figura 17.
El ID del usuario: string
El número total de tareas del usuario: int
El número total de tareas acabadas por el usuario: int
El número total de tareas en progreso del usuario: int
El número total de tareas en espera del usuario: int
El número total de tareas pendientes del usuario: int
Un array con todos los ids de las tareas en progreso o pendientes: array Int
Notificaciones recibidas de otros usuarios: JSON

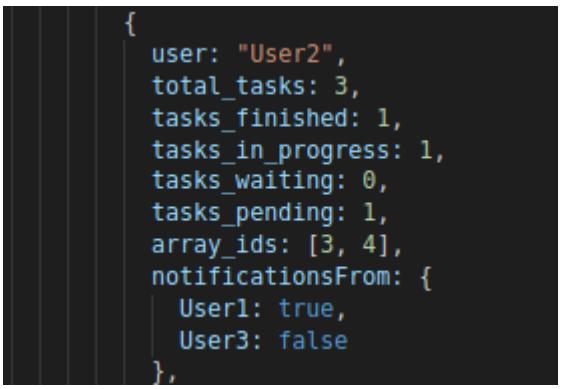


Figura 17. Definición de usuario

La **información general** del ledger, ver figura 18.
El número de usuarios: int
El número de tareas totales: int

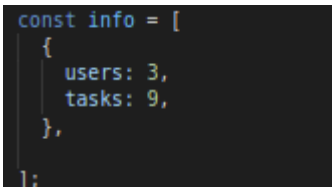


Figura 18. Definición general

Las funciones del contrato sirven para realizar transacciones sobre los activos o assets.
La **creación** de activos únicamente estará reservada para las tareas, es decir, cada usuario podrá crear tareas que deberán aprobar el resto de participantes de la red para ser añadidas a la base de datos.

La **consulta** de los activos se aplicará a todos, a las tareas, usuarios e información general. Todos los participantes podrán realizar estas consultas.

La **eliminación** de no está disponible en esta aplicación ya que no tiene sentido eliminar usuarios o tareas.

La **modificación** es donde se concentra gran parte de las funciones, se podrá modificar las tareas, usuarios e información general.

### 8.6.5. Diagrama de clases

A continuación en la figura 19 se especifica las diferentes clases que se contemplan en el proyecto.

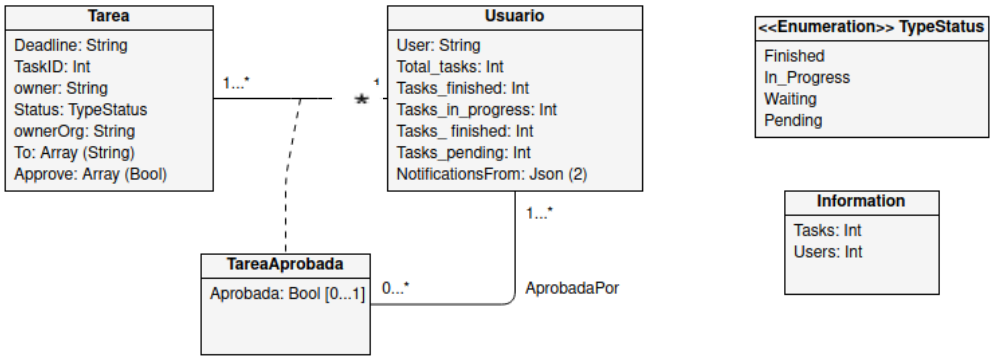


Figura 19. Diagrama de clases

### 8.6.6. Diagramas de secuencia

En la figura 20 se observa el diagrama de secuencia para métodos GET al ledger.
En la figura 21 se observa el diagrama de secuencia para métodos POST al ledger.

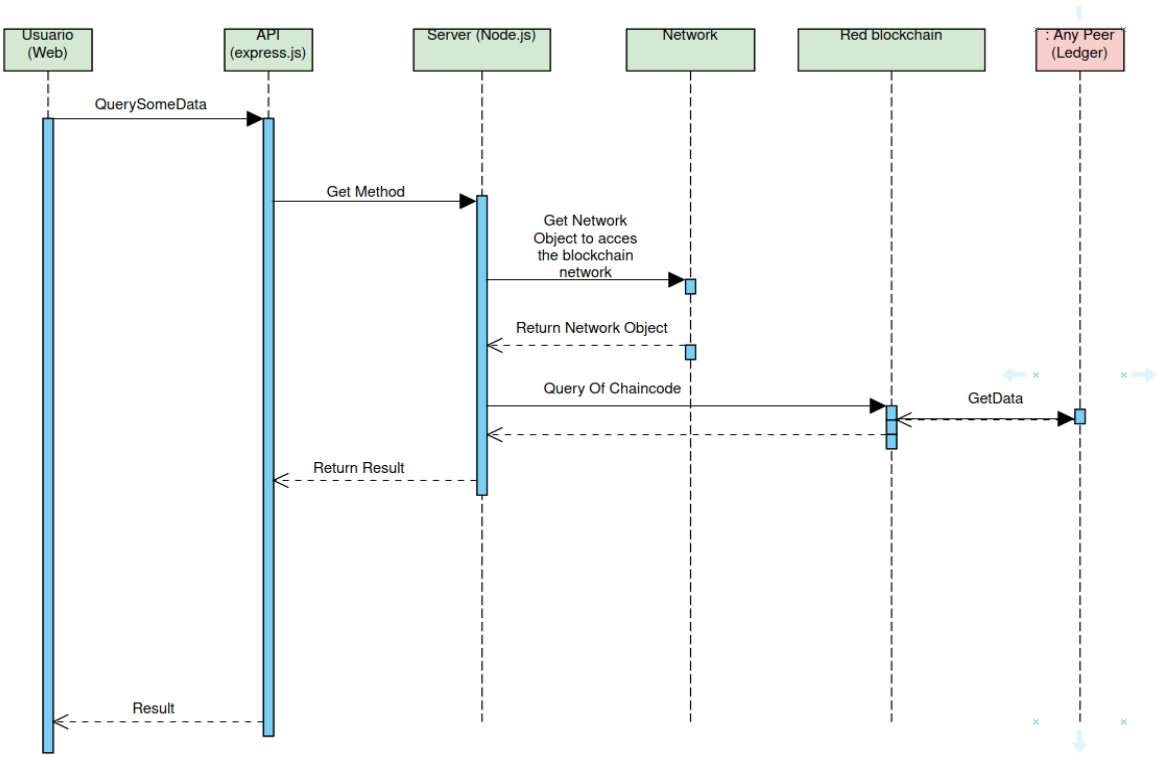


Figura 20. Diagrama de secuencia 1

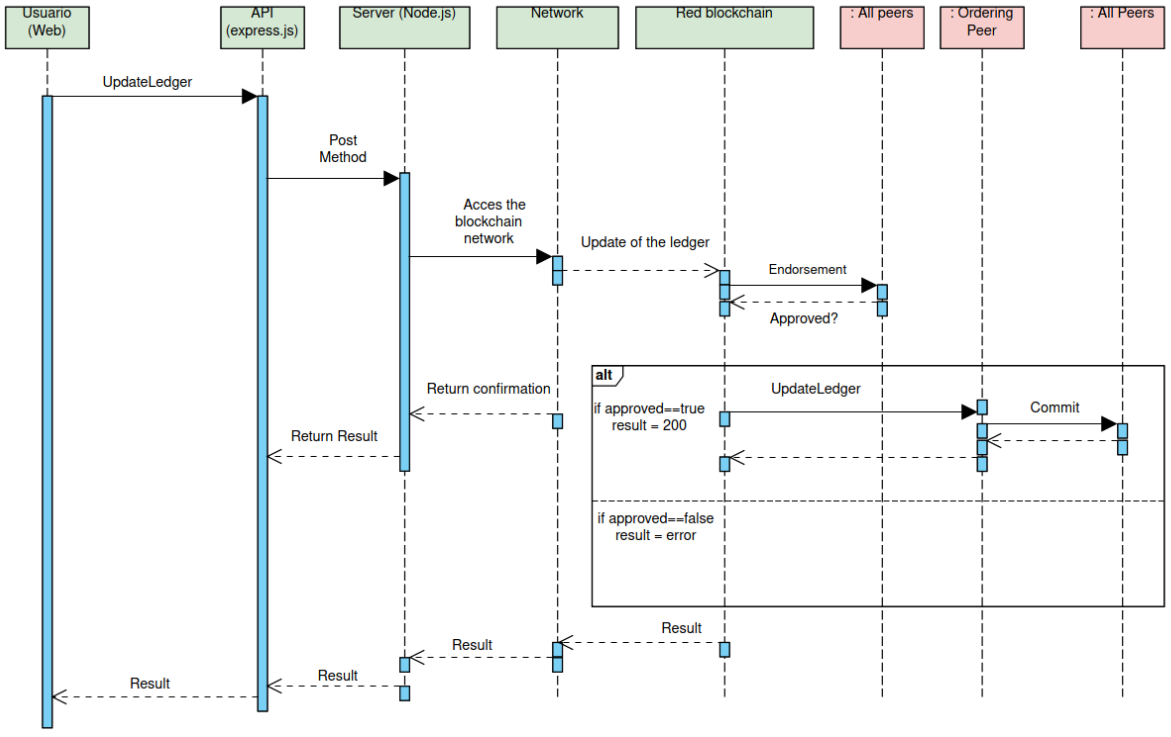


Figura 21. Diagrama de secuencia 2

### 8.6.7. Diagrama de estados de una tarea

En la figura 22 se observa los estados de una tarea desde su creación hasta su finalización.

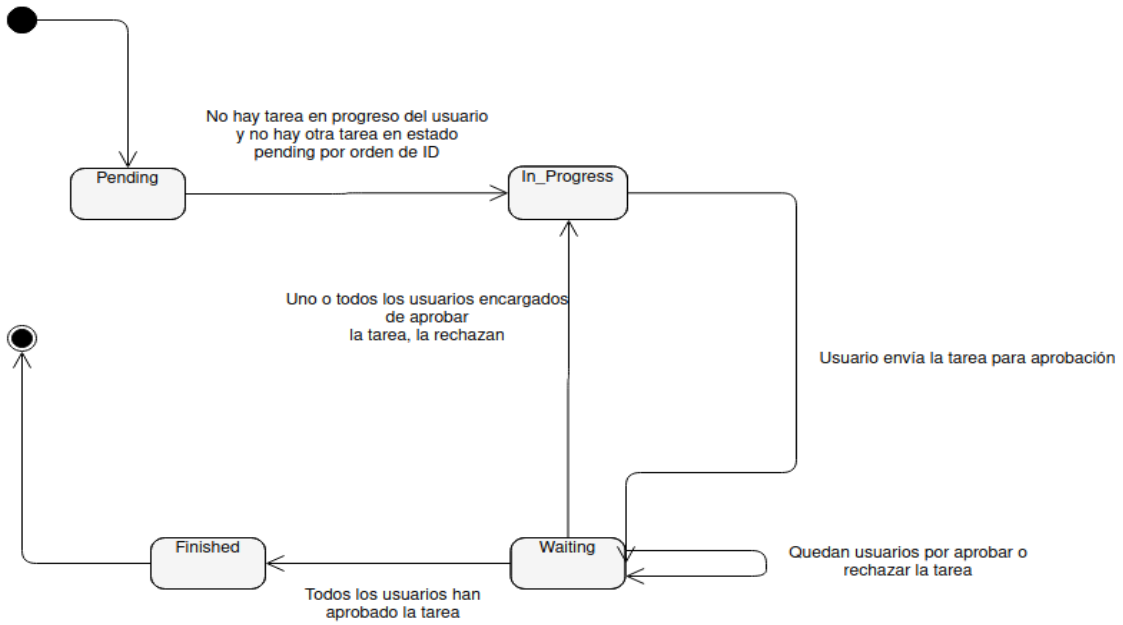


Figura 22. Diagrama de estados

El diagrama de estados se basa en las interacciones explícitas de los usuarios, es decir, no incluye el consenso de la red blockchain. Este consenso es automático e indica si la transacción es válida, ordena las transacciones y permite acordar un nuevo estado del libro de cuentas (ledger). Dependiendo de la naturaleza de la aplicación distribuida, por ejemplo si trabaja con sensores e IoT no hará falta una interacción con usuarios a lo largo de la vida del activo, en este caso, la tarea.

### 8.6.8. Base de datos donde se guarda la información: CouchDB

Para poder utilizar couchdb habrá que iniciar un contenedor docker para cada nodo que se crea y con una imagen couchdb, en este caso se usará la 3.1.1.

Una vez desplegado el contenedor se puede acceder a él mediante un buscador a la dirección: localhost:5984 (ver figura 23). Será la instancia de cada uno de los

contenedores y la información del libro de cuentas (ver figura 24) en formato JSON (ver figura 25).

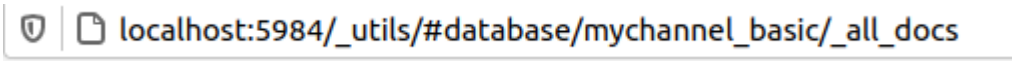


Figura 22. URL de la instancia couchdb

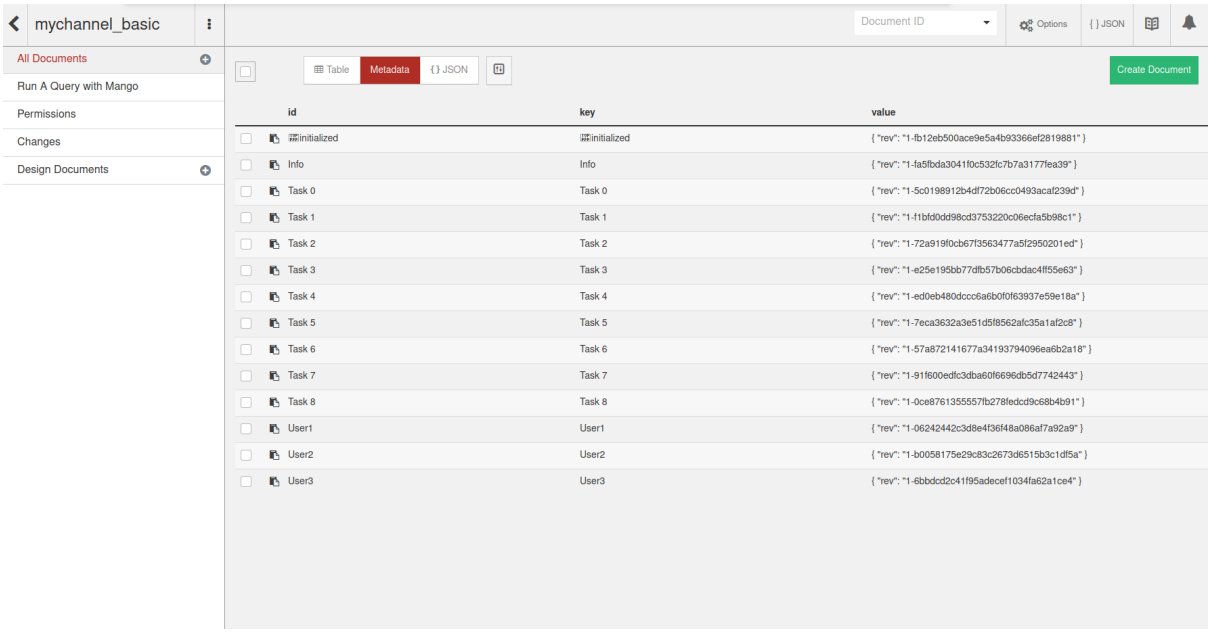


Figura 24. Información del ledger

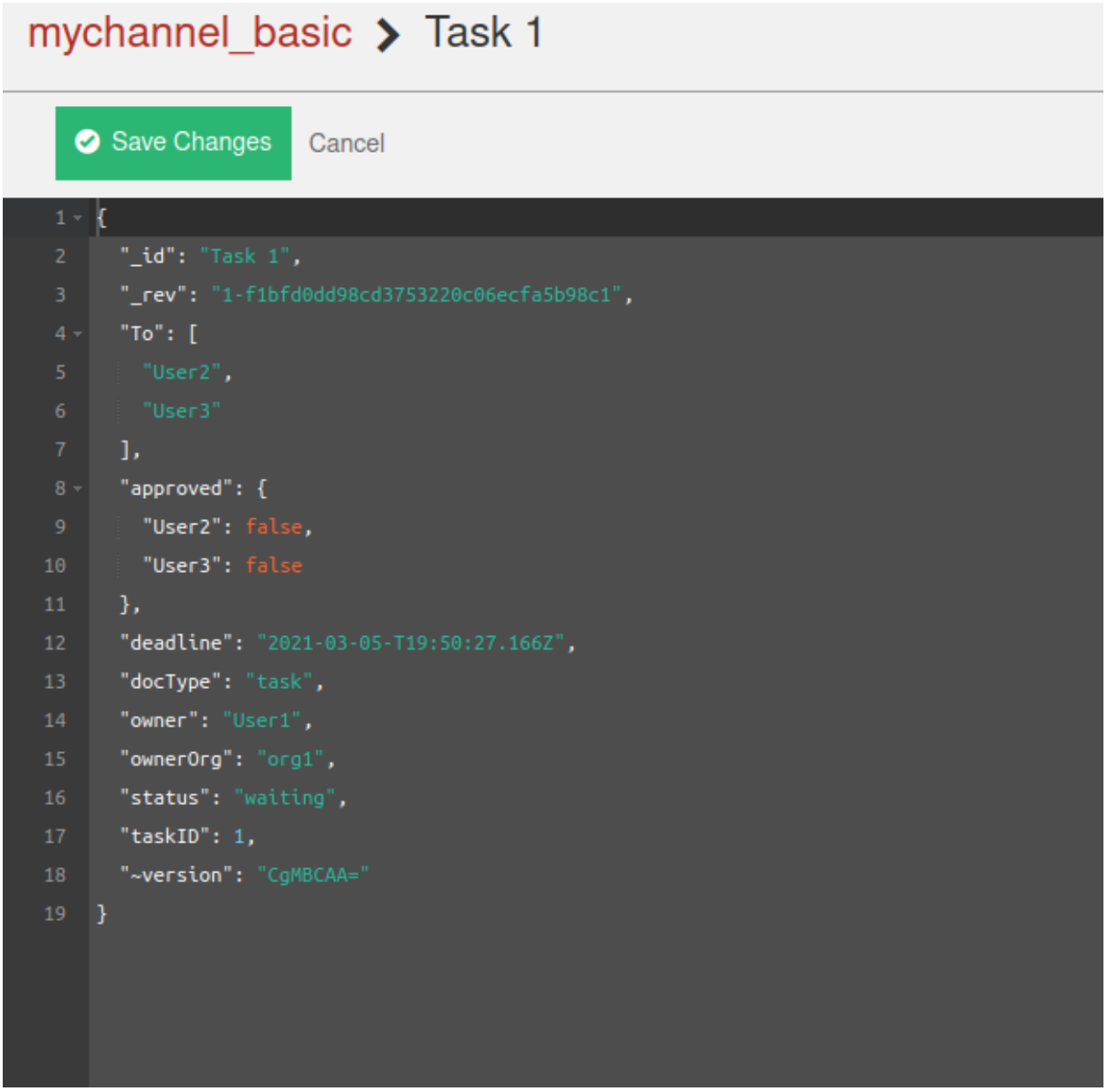


Figura 25. Elemento del ledger en formato JSON

## 8.7. Implementación

### 8.7.1. Tecnologías y lenguajes utilizados

Hyperledger Fabric samples: repositorio de scripts y ejemplos de fabric para desplegar componentes y testear chain codes.

Javascript: es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Nodejs: es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript, asíncrono, con E/S de datos en una arquitectura orientada a eventos y basado en el motor V8 de Google.

Vue.js: es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página.

Express.js: es un marco de aplicación web de back-end para Node.js está diseñado para crear aplicaciones web y API.

HTML: HyperText Markup Language, hace referencia al lenguaje de marcado para la elaboración de páginas web.

CSS: es un lenguaje de diseño gráfico para definir y crear la presentación de un documento estructurado escrito en un lenguaje de marcado

Bootstrap: Bootstrap es una biblioteca multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web.

### 8.7.2. Herramientas de desarrollo

Hyperledger Fabric

Visual Studio Code: Es un editor de código fuente desarrollado por Microsoft para Windows, Linux y macOS.

Json-server: Es una herramienta para desplegar servidores locales JSON y probar los archivos javascript con una URL

Github: Es una plataforma de alojamiento de código y permite la gestión de versiones en colaboración.

Git: Sistema de control de versiones por comando.

Postman: Es una plataforma que ayuda en la creación y pruebas de API.

Google Drive: Servicio de alojamiento de google.

Gmail: Servicio de correo de google.

Docker: Herramienta que automatiza el despliegue de aplicaciones dentro de contenedores.

### 8.7.3. Configuración y creación de la red

La configuración se basará en los requisitos de Hyperledger Fabric, otros modelos de redes privadas con permisos requerirán otra configuración y componentes.

#### Certificados

Cada organización que forme parte de la red deberá tener una autoridad de certificación que expida los certificados necesarios a:

Los integrantes de la organización que realizaran transacciones
Los peer / nodos que aprobaran las transacciones y guardarán la información
Los peer / nodos de ordenamiento que serán los que una vez aprobada las transacciones de actualización serán los que decidirán cómo se guardará la información en

Al no disponer de una autoridad de certificación utilice el Fabric-CA que permite crear servidores y clientes de certificación simulando la expedición de certificados.

Además de los certificados de identificación de usuarios, también se necesitan certificados TLS para asegurar la conexión y comunicación de los diferentes peers.

#### Contenedores Docker [18]

Para todos los contenedores se utiliza la imagen “latest” de Fabric que corresponde a la versión 2.3 y que inicializa todos los peers y autoridades de certificación con la información necesaria.

### Docker para autoridad de certificación

Los contenedores docker harán de autoridades de certificación y deberá haber una por organización. Como mi red consta de tres organizaciones se crearán tres contenedores para las autoridades. Además los nodos de ordenamiento deberán tener una autoridad, por lo que serán un total de cuatro autoridades de certificación.
En el fichero docker-compose-ca.yaml se puede encontrar la configuración de los CA, con los volúmenes y puertos asignados de mi máquina así como los comandos que inicializará el contenedor como servidor de tal manera que pueda expedir certificados.

Una vez creados los contenedores se generan los certificados registrando e inscribiendo a los componentes, como los peer, user1, admin y generando los certificados correspondientes. La lista de comandos se puede ver en el script registerEnroll.sh
En ccp-generate.sh está la configuración que hay que generar para poder conectarse a la red a partir de un gateway, este gateway se puede construir desde el servidor con express.js.

### Docker para organizaciones

Una vez creados todos los certificados necesarios se procede a crear los contenedores que harán el papel de los peers / nodos de las organizaciones, habrá uno por organización como mínimo y también un contenedor de ordenamiento.
En el fichero docker-compose-test-net.yaml se puede encontrar la configuración de variables, los puertos asignados y los comandos de inicialización de los peer y orderer.

### 8.7.4. Creación de canal de la red

La creación de la red la realiza un peer que esté autorizado por certificados, creados anteriormente. En este caso como se trabaja con un consorcio y se ha instalado los certificados en los tres peers, cualquiera de ellos puede crear la red. La lista que conforma el consorcio está guardada en los peers de ordenamiento y solo el admin del servicio de ordenamiento puede editar esta lista o listas.
Existen dos ficheros de configuración necesarios para la creación de la red. El .tx y un .block que en este caso serán mychannel.tx y mychannel.block.

mychannel.tx: Esta es una transacción de configuración que se enviará al servicio de ordenamiento, de modo que creará un nuevo canal y devolverá el bloque de génesis para el nuevo canal para que los peers puedan usarlo para unirse a él.

mychannel.block: El bloque que devuelve la transacción anterior y que contiene la configuración inicial del canal.
El mychannel.block lo pueden utilizar los peers para unirse al canal recién creado, por lo tanto, los tres peers de org1, org2, org3 deberán unirse al canal.

Por último se genera la configuración de anclaje.
Un peer en un canal que todos los demás pueden descubrir y que permita comunicarse se denomina peer de anclaje. Cada miembro (organización) de un canal tiene un peer de anclaje (o varios peers de anclaje para evitar un solo punto de falla), lo que permite que los peers que pertenecen a diferentes miembros descubran todos los pares existentes en un canal.

### 8.7.5. Despliegue del chaincode

Se requiere de una serie de pasos:

1. Empaquetar el contrato en un tar.gz
2. Instalar el contrato en un peer de cada organización
3. Cada peer deberá aprobar la definición del chaincode
4. El resto de peers deberán comprobar que el chaincode ha sido aprobado por todos los peers de la red
5. Una vez haya sido aprobado por todos los peers necesarios, se realizará el commit a la red.
6. Y una vez se haya realizado el commit se podrán hacer peticiones desde cualquier peer de las organizaciones que estén conectadas a la red.

## 8.8. Ejemplo de funcionamiento

### 8.8.1. Registro de usuario en la app

Primeramente habrá que registrar un usuario con un ID, contraseña y organización a la que pertenezca (Figura 26) y se podrá ver como en el wallet se crean sus credenciales (Figura 27). Para agilizar el proceso se han creado unos usuarios predeterminados en el smart contract así como sus tareas, que se inicializan en el ledger al desplegar la red.

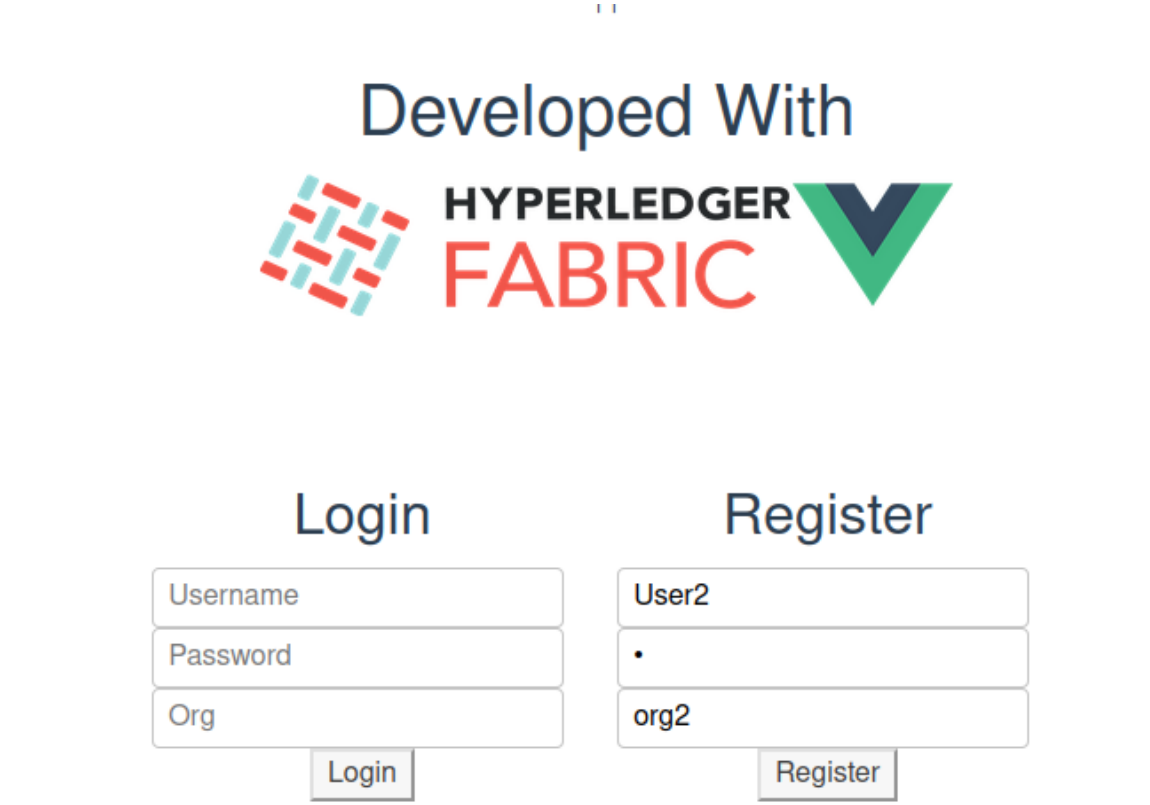


Figura 26

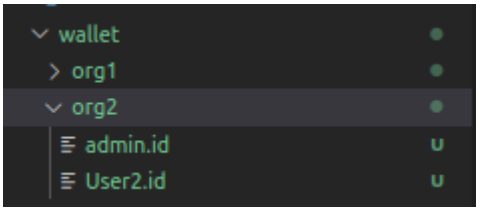


Figura 27

### 8.8.2. Login de usuario nuevo

Luego se procede a hacer login del usuario creado (Figura 28)

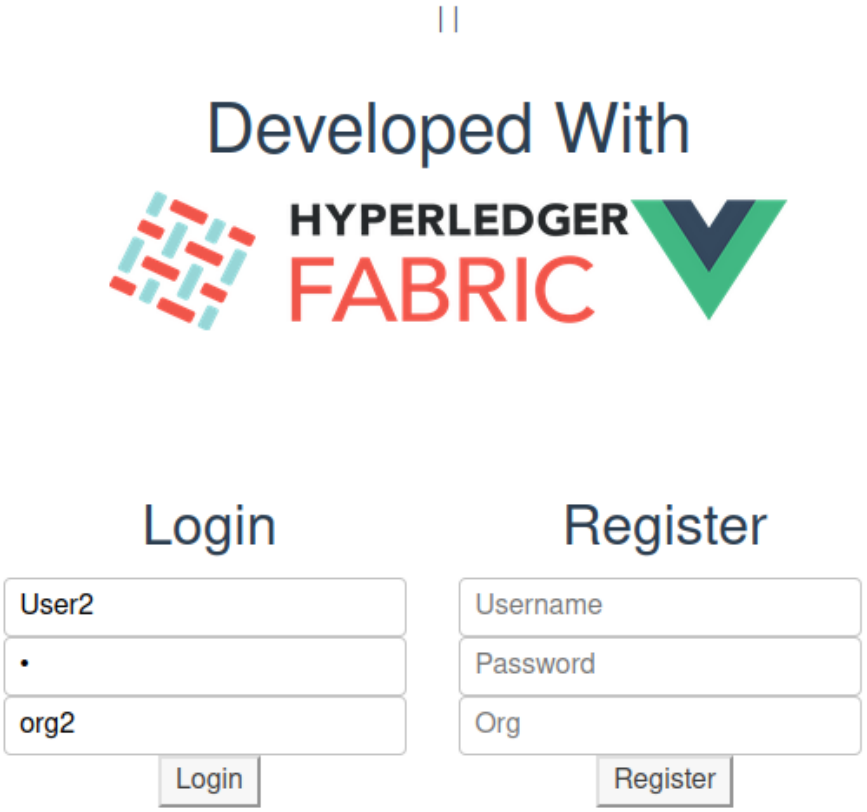


Figura 28

### 8.8.3. Listar tareas

Como se ha comentado anteriormente tanto los usuarios como las tareas se han guardado en el ledger al desplegar la red por lo que se pueden visualizar esas tareas en alguna de las instancias couchDB que proporciona la red (Figura 29, 30). También se podrán visualizar desde la interfaz de usuario que se inicia una vez se hace login de usuario (Figura 31).

| id | key | value |
|---|---|---|
| initialized | initialized | { "rev": "1-fb12eb500ace9e5a4b93366ef2819881" } |
| Info | Info | { "rev": "1-fa5fbda3041f0c532fc7b7a3177fea39" } |
| Task 0 | Task 0 | { "rev": "1-5c0198912b4df72b06cc0493acaf239d" } |
| Task 1 | Task 1 | { "rev": "1-f1bfd0dd98cd3753220c06ecfa5b98c1" } |
| Task 2 | Task 2 | { "rev": "1-72a919f0cb67f3563477a5f2950201ed" } |
| Task 3 | Task 3 | { "rev": "1-e25e195bb77dfb57b06cbdac4ff55e63" } |
| Task 4 | Task 4 | { "rev": "1-ed0eb480dccc6a6b0f0f63937e59e18a" } |
| Task 5 | Task 5 | { "rev": "1-7eca3632a3e51d5f8562afc35a1af2c8" } |
| Task 6 | Task 6 | { "rev": "1-57a872141677a34193794096ea6b2a18" } |
| Task 7 | Task 7 | { "rev": "1-91f600edfc3dba60f6696db5d7742443" } |
| Task 8 | Task 8 | { "rev": "1-0ce87613555571b278fedcd9c68b4b91" } |
| User1 | User1 | { "rev": "1-06242442c3d8e4f36f48a086af7a92a9" } |
| User2 | User2 | { "rev": "1-b0058175e29c83c2673d6515b3c1df5a" } |
| User3 | User3 | { "rev": "1-6bbdcd2c41f95adecef1034fa62a1ce4" } |

Figura 29


```
[
  {
    Key: 'Task 0',
    Record: {
      To: [Array],
      approved: [Object],
      deadline: '2021-11-03-T19:50:27.166Z',
      docType: 'task',
      owner: 'User1',
      ownerOrg: 'org1',
      status: 'finished',
      taskID: 0
    }
  },
  {
    Key: 'Task 1',
    Record: {
      To: [Array],
      approved: [Object],
      deadline: '2021-03-05-T19:50:27.166Z',
      docType: 'task',
      owner: 'User1',
      ownerOrg: 'org1',
      status: 'waiting',
      taskID: 1
    }
  },
  {
    Key: 'Task 2',
    Record: {
      To: [Array],
      approved: [Object],
      deadline: '2021-01-04-T19:50:27.166Z',
      docType: 'task',
      owner: 'User2',
      ownerOrg: 'org2',
```


Figura 30

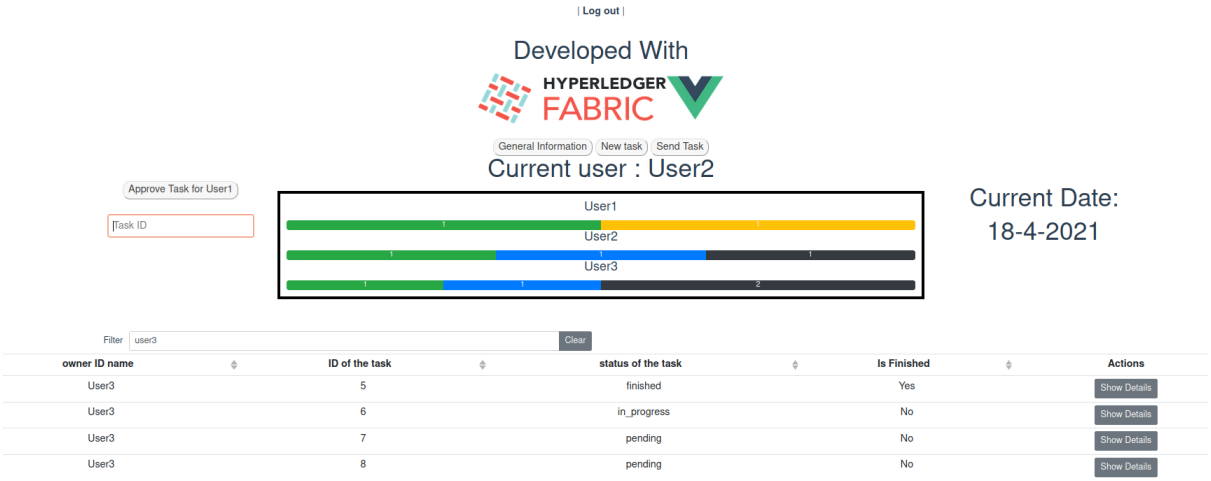


Figura 31

## 8.8.4. Progreso de tareas

Al hacer login se observará una tabla de progresos de colores que indicará el estado de cada una de ellas (Figura 32). Se podrá consultar las tareas con un buscador para conocer sus detalles (Figura 33).

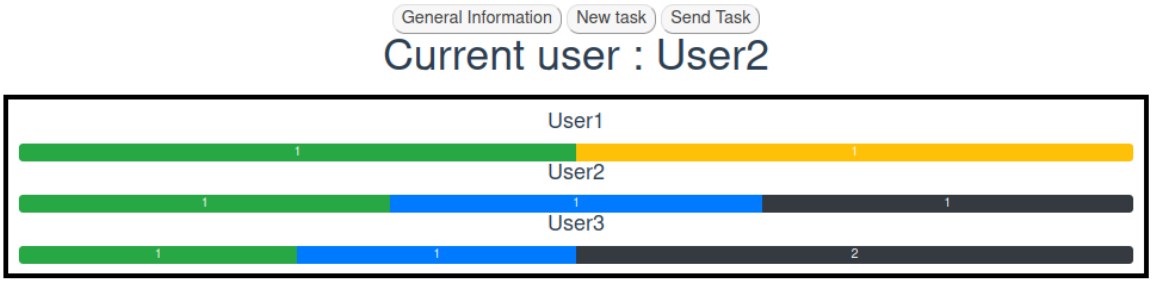


Figura 32

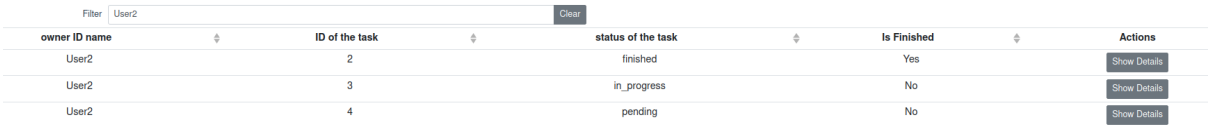


Figura 33

## 8.8.5. Ampliación con una tarea

Además de las tareas preestablecidas se podrán crear tareas nuevas clicando en el botón de 'New Task' (Figura 34). Al hacer click nos redirigirá a una ventana en la cual se tendrá que introducir los detalles de la nueva tarea a crear (Figura 35).

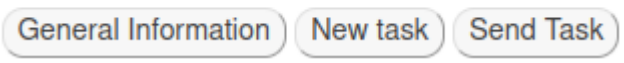


_Figura 34

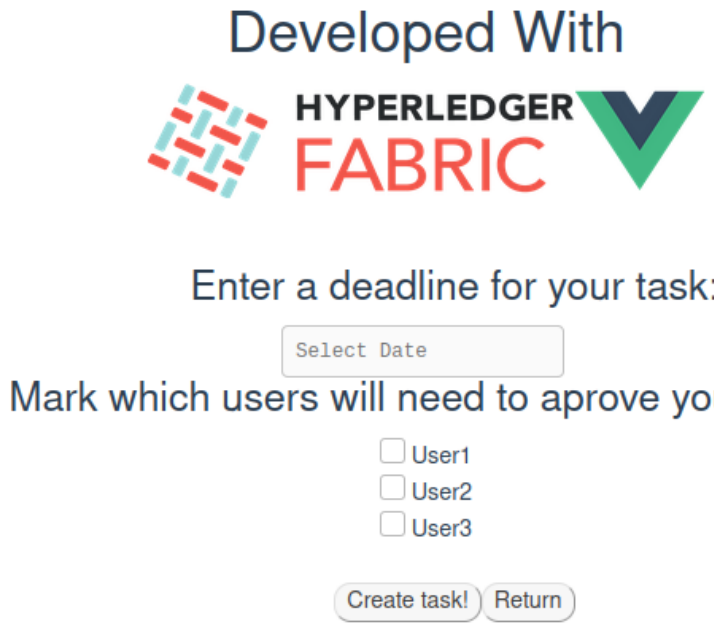


_____Figura 35

Para crear la tarea se debe escoger una fecha que será la fecha de fin de la tarea y los usuarios que deberán aprobar antes de ser finalizada (Figura 36). Y darle al botón de 'Create task!'.

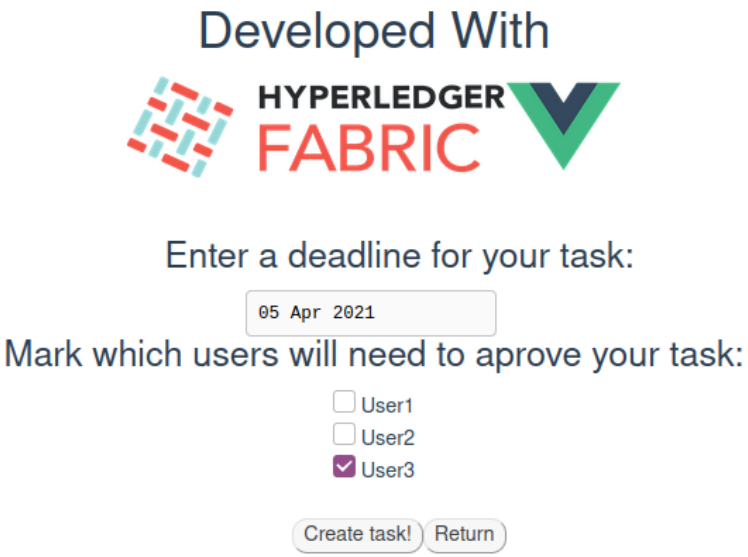


Figura 36

### 8.8.6. Cómo se registra la tarea en el ledger

Una vez se rellenan los campos y clica en el botón de crear la tarea, el front end inicia una función que envía una POST request al servidor (server.js) express mediante un servicio de API (apiservice.js). El servidor inicia la conexión con la red, en la cual se harán las comprobaciones con el wallet para confirmar que el usuario que se conecta es legítimo. Una vez las comprobaciones hayan terminado se iniciará la transacción con el smart contract y se registrará en el ledger mediante el protocolo de consenso Raft (Ver figuras 37, 38, 39) que creará el nuevo bloque con la transacción de la tarea nueva que corresponde al bloque 9.


Figura 37: Logs de peer 3

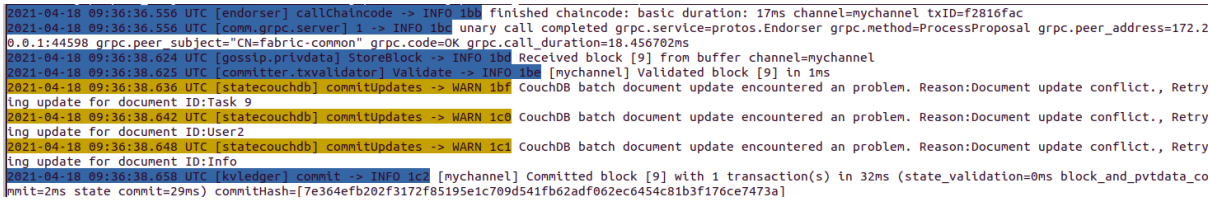


Figura 38: Logs de peer 2

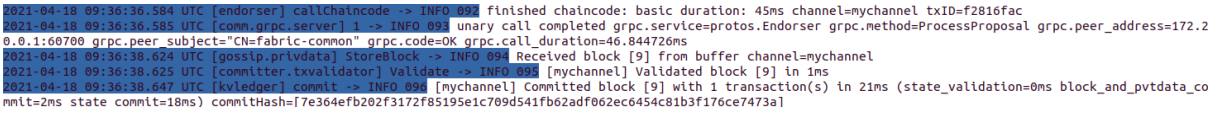


Figura 39: Logs de peer 1

### 8.8.7. Visualización de la nueva tarea

Una vez guardada la tarea en el ledger se podrá visualizar desde la base de datos couchDB (Figura 40, 41) y desde la aplicación (Figura 42, 43).

| id | key | value |
|---|---|---|
| initialized | initialized | {"rev": "1-fb12eb500ace9e5a4b93366ef2819881"} |
| Info | Info | {"rev": "2-bbf050dfd14453eaeaf772f18ae7eafe"} |
| Task 0 | Task 0 | {"rev": "1-58a7faf6d8bb311ede53cafc01e2f963"} |
| Task 1 | Task 1 | {"rev": "1-b13d8258de6986dd3b0c777f3e64b1dd"} |
| Task 2 | Task 2 | {"rev": "1-360dd27c35a91c230281a061beebf766"} |
| Task 3 | Task 3 | {"rev": "1-ff4dd9a44149592d8a32e2f2580b96d2"} |
| Task 4 | Task 4 | {"rev": "1-0fe17c6f3368f143df2dc03e0a5e758d"} |
| Task 5 | Task 5 | {"rev": "1-9b152d849247bac184f0ecc2e64cf677"} |
| Task 6 | Task 6 | {"rev": "1-b60f855b9d3d10c05bbc7a3e267cd5bb"} |
| Task 7 | Task 7 | {"rev": "1-1b7ae1a5996194a57174801e4e1eb179"} |
| Task 8 | Task 8 | {"rev": "1-9c3ed2124725783107ea8694853ca9a1"} |
| Task 9 | Task 9 | {"rev": "1-526807b2e3bfcc405fc65d2c76c9b1df"} |
| User1 | User1 | {"rev": "1-3b20ee33038d53dcfb4450f6cb50cf97"} |
| User2 | User2 | {"rev": "2-29a19d0c710887672e295cbde5ae0ac0"} |
| User3 | User3 | {"rev": "1-23ca9fc6a929fabb8912c0f516647b97"} |

Figura 40


```
{
  "_id": "Task 9",
  "_rev": "1-526807b2e3bfcc405fc65d2c76c9b1df",
  "To": [
    "User3"
  ],
  "approved": {
    "User3": false
  },
  "deadline": "",
  "docType": "task",
  "owner": "User2",
  "ownerOrg": "org2",
  "status": "pending",
  "taskID": 9,
  "~version": "CgMBCQA="
}
```


Figura 41

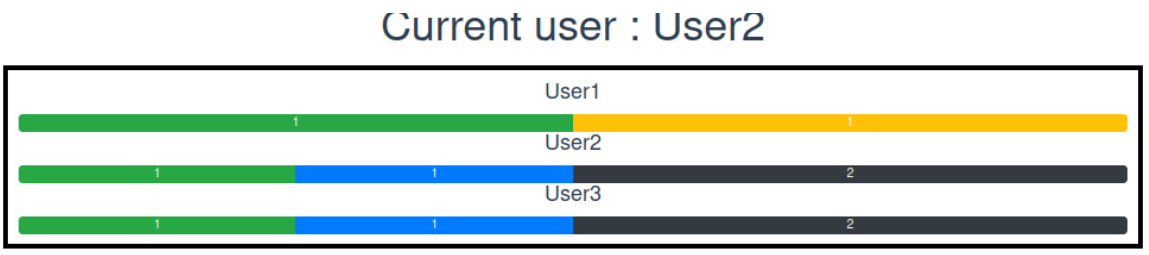


Figura 42

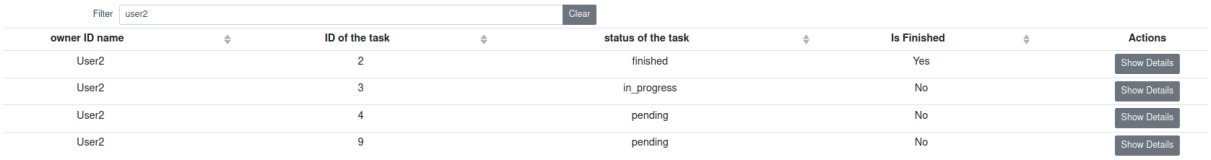


Figura 43

### 8.8.8. Enviar la tarea en progreso a aprobar

Si se tiene una tarea disponible para finalizar se clica en el botón de 'Send Task' y aparecerá un mensaje conforme se ha enviado la tarea a la blockchain para cambiar de estado (Figura 44). El estado de la tarea pasará a waiting (Figura 45, 46) y los usuarios que la deban aprobar deberán entrar a su cuenta para dar su consentimiento a la finalización de la tarea.
Se puede consultar qué usuarios deben aprobar la tarea con ayuda de la tabla de visualización (Figura 47).

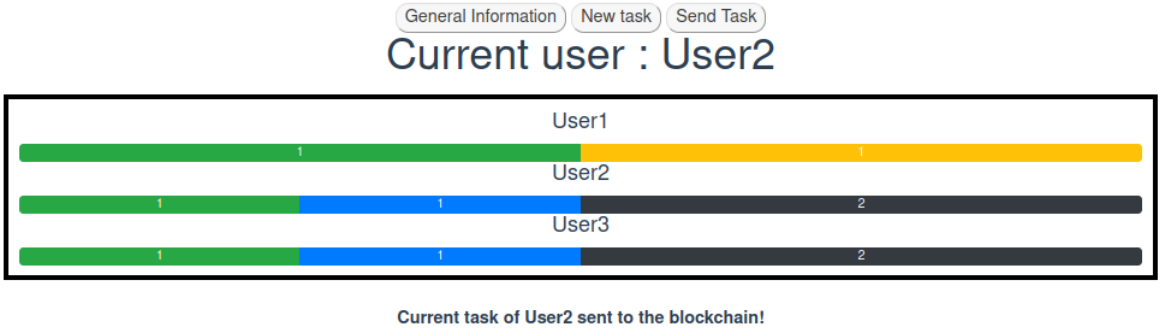


Figura 44

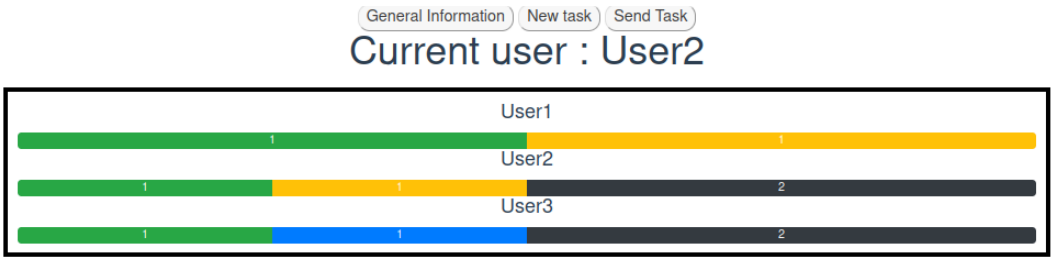


Figura 45

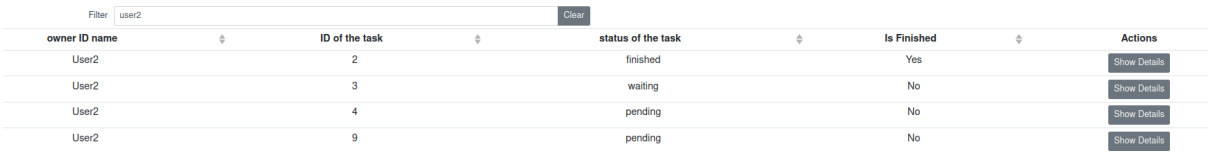


Figura 46


Figura 47

## 8.8.9. Ver las notificaciones recibidas a los usuarios correspondientes de la tarea para aprobar

Una vez el usuario que debe aprobar la tarea haga login, en este caso el 'User3', nos aparecerá en la esquina superior izquierda las notificaciones pendientes (Figura 48). Para aprobar la tarea se ha de escribir el ID de la tarea en el cuadro y darle al botón 'Approve Task for User2' (Figura 49).

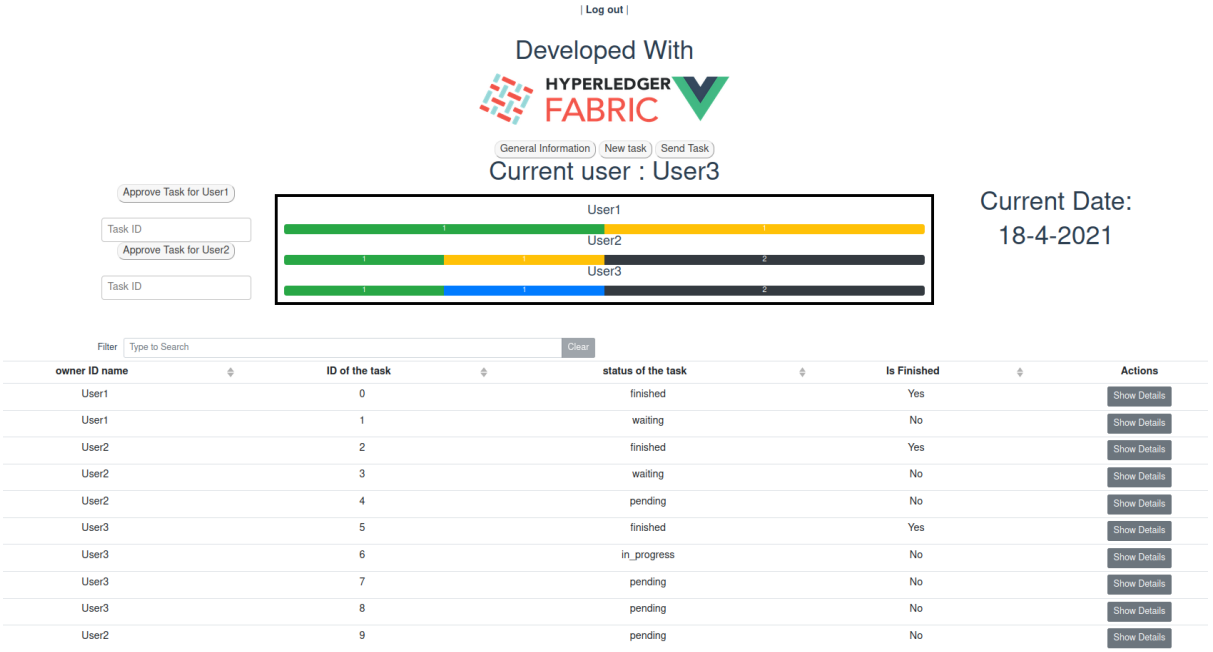


Figura 48

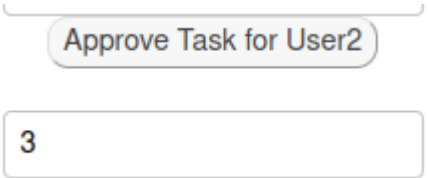


Figura 49

### 8.8.10. Listar las tareas de nuevo y comprobar la barra de progreso ha cambiado

Una vez se haya aprobado la tarea se podrá observar cómo la barra de progresos ha cambiado y la notificación del ‘User2’ ha desaparecido de la pantalla de inicio del ‘User3, en el cuadro de visualización también se observa el cambio de estado’ (Figura 50).

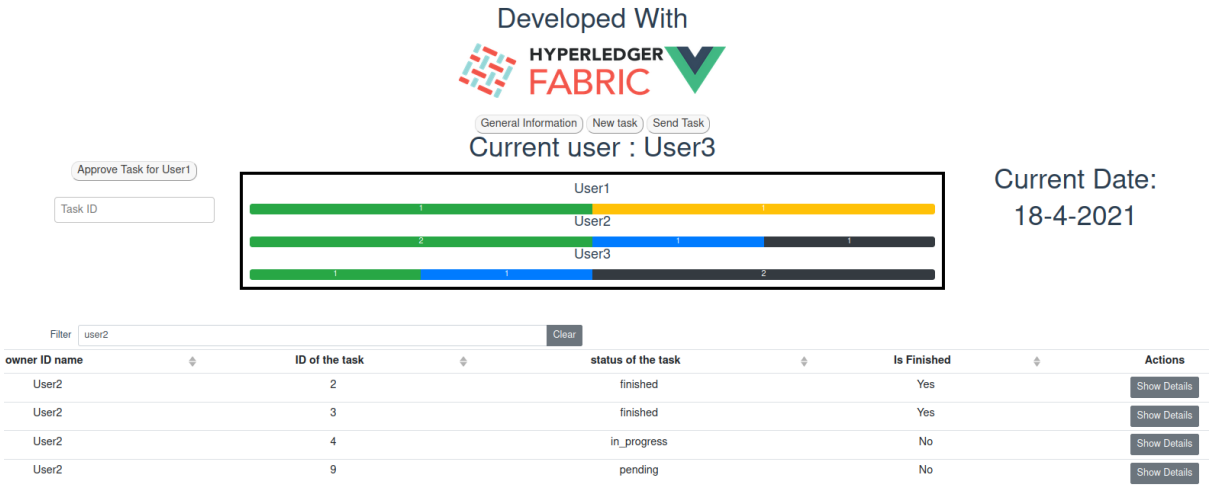


Figura 50

# 9. Conclusiones y trabajo futuro

Primeramente sobre la comparativa se obtiene que Hyperledger Fabric ha sido implementado específicamente para uso de empresa, a diferencia de Ethereum Quorum. Esto deriva en que el rendimiento de Fabric es muy superior ya que su arquitectura está pensada para satisfacer las necesidades de empresas, como la velocidad de transacción en lugar de la seguridad. Esto se ve en qué Fabric no requiere de encriptación de los datos, pero Quorum sí ya que los datos pueden ser visibles por entidades externas sin ser parte del grupo empresarial. Además el número de transacciones por segundo de Fabric (3000) son muy superiores a Quorum (100).

Ambas redes tienen características, componentes y funcionalidades más o menos complejas y por ello sería ideal poder integrar diferentes funcionalidades de diversas redes a otras, para que de esta manera se pueda potenciar su utilidad. O bien tener una manera sencilla de poder comunicar estas redes de manera que se pueda utilizar la infraestructura y componentes de diferentes redes.
Lamentablemente esto es algo que todavía no es posible y que habría que considerar en un trabajo futuro. Ya que a medida que esta tecnología cobra fuerza, van surgiendo diferentes redes con nuevas funcionalidades y tener que aprender a programar o configurar la red, para poder sacar el máximo provecho de la misma, requiere invertir esfuerzo y tiempo que puede llegar a no ser viable.

En cuanto a la aplicación, cumple el objetivo de validar la información recogida ya que se ha implementado con Hyperledger Fabric. También muestra la forma de automatizar una aplicación ya que toda la ejecución depende de la implementación del smart contract, de la comunicación con los peer y el servicio de ordenamiento. Si la comunicación con los peer falla, las llamadas realizadas desde el front-end no se llegarán a ejecutar y por lo tanto la información no será alterada, hasta que los peer vuelvan a estar disponibles.  Este tipo de red hace que no sea posible alterar el ledger si no se respeta la definición del smart contract y de tener la autorización/permisos que cada peer otorga proporcionando así una aplicación descentralizada otorgando fiabilidad, transparencia y seguridad.

# 10. Referencias

## Bibliografia

[1] Vitalik Buterin. *A next generation smart contract & decentralized application platform*, 2013

[2] Nick szabo. f*ormalizing and securing relationships on public networks,* 1997

[3] Shuai Wang, Yong Yuan, Xiao Wang Juanjuan Li, Rui Qin, Fei-Yue Wang. *An Overview of Smart Contract: Architecture, Applications, and Future Trends*, 2018

[4] Silas Nzuva *Smart Contracts Implementation, Applications, Benefits, and Limitations,* 2019

[5] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidhran, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrsoula Stahakopoulou, Marko Vokolic, Sharon Weed Cocco, Jason Yellic. *Hyperledger Fabric: A Distributed Operating System for Permissioned Blockchains*, 2018

[6] Stanislaw P. Stawicki, Michael Firstenberg, Thomas Jhon Papadimos. *What's new in academic medicine? Blockchain technology in health-care: Bigger, better, fairer, faster, and leaner, 2018*

[7] Svein Olnes, Jolien Ubatch, Marijin Janssen. *Blockchain in government_ Benefits and implications of distributed ledger technology for information sharing* From *Government Information Quarterly Book.*

/10

## Webgrafia

[8] Documentación Solidity: https://solidity-es.readthedocs.io/es/latest/

[9] Hyperledger fabric: https://www.hyperledger.org/use/fabric

[10] Ethereum Quorum https://consensys.net/quorum/developers/

[11] IBM Developing smart contracts with IBM Blockchain Platform Developer Tools

https://cloud.ibm.com/docs/blockchain-sw-25?topic=blockchain-sw-25-develop-vscode

[12] https://www.oracle.com/es/application-development/cloud-services/blockchain-platform/

[13] https://docs.microsoft.com/es-es/azure/blockchain/service/

[14] https://hyperledger-fabric.readthedocs.io/en/release-2.2/whatsnew.html

[15] Documentación para Vue.js https://vuejs.org/v2/guide/components.html

[16] Documentación sobre expressjs https://expressjs.com/

[17] Herramienta para crear diagramas https://online.visual-paradigm.com/

[18] Documentación docker https://docs.docker.com/

[19] Página de consultas sobre errores https://stackoverflow.com/