UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA DEL SOFTWARE PARA LA WEB

Trabajo Fin de Máster

MANTENIMIENTO INDUSTRIAL Y TRAZABILIDAD

Jose María de la Torre Antolín

2023

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN

INGENIERÍA DEL SOFTWARE PARA LA WEB

Trabajo Fin de Máster

"MANTENIMIENTO INDUSTRIAL Y TRAZABILIDAD"

Autor: **Jose María de la Torre Antolín**

Director: **Jose María Gutiérrez Martínez**

Tribunal:

Presidente: .....

Vocal 1º: ..... Vocal 2º: .....

Calificación: .....

Fecha: ..... de ..... de .....

ÍNDICE RESUMIDO

ÍNDICE DETALLADO

INTRODUCCIÓN

En la industria es frecuente externalizar servicios por distintas razones: excesiva demanda para los recursos disponibles, diversificación del riesgo, estrategia comercial, son ajenos al negocio principal, etc. El transporte de productos, asesoría fiscal y laboral, calibración de máquinas o herramientas, renting de vehículos y un largo etcétera son ejemplos de servicios que las organizaciones encargan a otras para realizarlos.

Algunos de estos servicios como puede ser el transporte de productos o la calibración de máquinas o herramientas se exige además que se realice con algún tipo de trazabilidad en su ejecución. Así puede ser necesario comunicar en tiempo real los distintos pasos geográficos de las mercancías transportadas, incluso quizás asegurar el mantenimiento de cierto margen de temperatura durante el transporte o el mantenimiento de un registro con las calibraciones realizadas a los indicadores de temperatura del camión como forma de asegurar que continúan cumpliendo con las características descritas por el fabricante.

Este proyecto trata de ofrecer una solución a un caso de uso de este tipo, como es el registro de la externalización de servicios de mantenimiento de maquinaria, equipos y herramientas, desde el punto de vista de una organización que pertenece al sector aeronáutico. Esta industria se diferencia por el exigente grado de aseguramiento de la calidad aplicado. Se identifica, documenta y registra todos los aspectos que la rodean: la formación y capacitación del personal, cada herramienta cuando y donde fue utilizada, la permanente actualización de los procesos de trabajo descritos por el fabricante o los propios de la organización, el origen de los repuestos, su certificación, compatibilidad y donde fueron utilizados, auditorías periódicas internas y externas, exhaustivo mantenimiento preventivo de todas las partes de la aeronave y un largo etcétera que busca reducir al máximo la posibilidad de que una cadena de errores lleve a un fallo catastrófico en máquinas tan complejas.

Dentro de este universo de relaciones tan complejo, este trabajo utiliza el caso de uso descrito para ejemplificar uno de este tipo de sistemas:

- un sistema interno a la organización que registra y documenta todos sus procesos, representado por la gestión burocrática llevada a cabo para externalizar servicios de mantenimiento a otros proveedores. Es un sistema que incluye un sistema de almacenamiento de información tradicional.

Gestión de Externalización Servicios de Mantenimiento.

La gestión de los servicios de mantenimiento externalizados ejemplifica de alguna forma a todos los procesos internos a la organización aeronáutica. La informatización de esta gestión, registra todo el ciclo de vida de una petición de mantenimiento externo a un proveedor de este servicio, como puede ser la identificación de la propia petición, el material afectado, la operación de mantenimiento solicitada, su precio, proveedor que realiza la petición, cuando el material es entregado al proveedor, cuando lo devuelve, el estado final del material, si se ha facturado o no, etc.

Registro de la operación de mantenimiento.

El proveedor que va a realizar el trabajo es el responsable de registrar en su propio sistema todo el ciclo de vida que sigue el material para que la operación de mantenimiento contratada se lleve a cabo como por ejemplo cuando el proveedor recoge el material, operación de mantenimiento contratada, operaciones finalmente realizadas, manuales aplicados, repuestos empleados, etc.

El aseguramiento de la calidad del trabajo realizado está garantizado pues el proveedor está obligado legalmente a mantener un estándar de calidad específico para poder ofrecer sus servicios.

Por simplicidad se ha supuesto que el transporte de ida y vuelta es realizado por la propia empresa contratista del servicio.

La operación de mantenimiento encargada a un proveedor, es un ejemplo de todos los procesos que sigue la organización aeronáutica en su relación con distintos proveedores, e incluso, algunos colaboran entre ellos para llevar a cabo dichos servicios. El sistema informático utilizado para gestionar sus interrelaciones podría ser, en última instancia, compartido por todos ellos. Para ejemplificar este otro tipo de sistema, que interconecta con los proveedores de servicios este trabajo propone:

- un sistema compartido entre clientes y proveedores, externo (de alguna forma) a ambas organizaciones, pero compartido por todas ellas, cuya función sea el registro de los servicios proporcionados por los proveedores, representado por el registro de las operaciones de mantenimiento realizadas a solicitud de una organización cliente. Es un sistema que debe incluir un sistema de almacenamiento de información inalterable y con un alto nivel de seguridad, pues guarda información muy sensible, ligada al extremo nivel de calidad de los trabajos que se realizan en determinadas industrias.

OBJETIVOS DEL PROYECTO

El objetivo del este proyecto es desarrollar una aplicación que realice el control de la gestión de peticiones de mantenimiento, envío del material a reparar, recepción y facturación con la empresa contratada.

Las comunicaciones previas entre el centro reparador y la organización cliente para llegar a un acuerdo sobre los términos del servicio, se realizan por correo electrónico y teléfono.

En la actualidad este trabajo se realiza con hojas de cálculo tipo Excel y se busca centralizar la información y estandarizar el registro del proceso de externalización.

Como segundo objetivo se realizará una investigación sobre la tecnología blockchain aplicada al problema de la calidad de la información compartida entre organizaciones y su confianza en ella. El objetivo es desarrollar un segundo sistema, que implemente una red blockchain, de la que tanto el proveedor como la organización cliente formen parte. El objetivo es que la organización cliente esté informada de lo que sucede con los equipos que envía mantener, a veces a otros continentes (cual es la situación del equipo, cómo avanza su reparación, su estado final, etc.) y que el proveedor vuelque en la red la información sobre los trabajos realizados. Para que ambas organizaciones confíen en la información, esta debe ser inalterable y segura.

Para la realización de este proyecto se han utilizado los conocimientos adquiridos en el máster, complementándolos con otras tecnologías como blockchain.

ESTADO DEL ARTE

1.1. Introducción.

Para liberar del trabajo repetitivo a las personas hace falta tecnología, sin embargo su uso excesivo puede resultar perjudicial para nuestro rendimiento en el trabajo. Este trabajo está redactado utilizando un procesador de textos. Un prodigio si se compara con los "escritos" tecleados en una máquina de escribir mecánica. Sin embargo existe una gran literatura acerca de como optimizar nuestro tiempo dedicado al trabajo, explicando técnicas de gestion para el uso del teléfono o el correo electrónico. Es preciso preguntarse hasta qué punto adoptar una nueva tecnología produce mejoras que realmente tienen un impacto positivo (Fenández, 2017) , o acaba resultando ser una moda más?

Este trabajo realiza una investigación, una prueba de concepto, para experimentar si realmente aporta valor a los sistemas informáticos de backoffice utilizar tecnología blockchain para el seguimiento del mantenimiento externo de la maquinaria productiva.

No es necesario un ladrón de tiempo más, pero no son necesarios los 70 años que se tardó en adoptar la hoja de cálculo (Fenández, 2017). No en un mundo empresarial en el que cualquier pequeña mejora tecnológica puede suponer una gran ventaja competitiva. En el caso de las hojas de cálculo, es tal el éxito alcanzado por este tipo de herramientas, que 40 años después los usuarios continúan sacándole partido de las formas más insospechada por su flexibilidad, sencillez y potencia.

Según el diccionario panhispánico del español jurídico, el término backoffice se refiere como "personal de una empresa que desarrolla las tareas administrativas asociadas a las operaciones de la misma y que no tiene trato directo con los clientes" (Real Academia Española de la Lengua y otros, 2022a). Por la naturaleza del diccionario, menos específico, el diccionario de Cambridge nos acerca más al campo que nos ocupa y define la palabra backoffice como "the part of a business company that is concerned with running the company and that does not deal directly with customers or the public" (Cambridge University Press, 2022) (la parte de una empresa que la administra pero no trata directamente con los clientes - traducción libre). Forman parte de las funciones del backoffice de las empresas el departamentos de recursos humanos, la gestion de la tesorería, gestiones administrativas como la elaboración de impuestos, licencias de exportación, cálculo de comisiones sobre ventas, la gestion de los almacenes o el mantenimiento de las máquinas que elaboran los productos de la empresa.

Estas funciones que, en general, apoyan el proceso productivo de una empresa pueden ser externalizadas y es otra organización la que las realiza. Es lo que se denomina outsourcing (Real Academia de la Lengua y otros, 2022b), existiendo toda un sector empresarial cuya actividad principal es ofrecer este tipo de servicios para otras empresas. (DirigentesDigital, 2015)

Hoy en día, para el adecuado desarrollo de todas estas funciones, la empresa se apoya en diferentes sistemas informáticos. A partir de un tamaño determinado las empresas necesitan un software de mucha mayor potencia que las hojas de cálculo que tanto valor les han aportado durante todos estos años. Necesitan sistemas informáticos que capturen la lógica de los procesos del negocio y den soporte temporal

a las conclusiones obtenidas en bases de datos. Del análisis posterior de esos datos pueden los directivos de las empresas obtener conclusiones que ayuden a guiar su acción.

Sin embargo, el campo de estudio de este trabajo es el que da soporte a necesidades de información que surgen en el devenir diario de las operaciones de la empresa. Sistemas que responden a la pregunta ¿qué ocurre con …., que lo necesito para….?, un instante después de formularla. Para poder responderla, se recurre a los sistemas informáticos que dan soporte a las funciones de backoffice de la empresa. Pero ¿qué ocurre si la pregunta está referida a una entidad que ha abandonado termporalmente la empresa con destino temporal a otra organización que se dedica a proporcionar servicios de outsourcing? ¿Qué sucede con productos entregados por proveedores que deben ser enviados a la empresa que proporciona el outsourcing? La entidad está fuera de la empresa. En sus sistemas solo figurarán estimaciones sobre el estado de la entidad situada fuera de la empresa, pero no certezas. Sin soporte informático de esta necesidad, la solución pasa por una gestion administrativa. Es necesario comunicarse con la organización externa a través de los medios de comunicación tradicionales, correo electrónico o teléfono, e informarse del estado de la entidad externalizado y actualizar manualmente la información en los sistemas de la empresa.

Un paso adelante hacia el objetivo de automatizar la actualización de la información, es recibir periódicamente la información desde la organización externa en un formato normalizado de datos, y a partir de él actualizar los sistemas de la empresa. Es posible por ejemplo, recibir una vez al mes por correo electrónico un archivo de hoja de cálculo con la relación de elementos externalizados y su estado en ese momento, e implementar un proceso que la integre de forma automática en los sistemas de la empresa.

A	B	C	E	F	G	H	I	J	L
							STATUS COMPONENTES		
Pedido Cliei	SN Nº	Fecha Av	Descripción S/N	Comp. S/N	A	Fecha Inicio	Fecha Fin A	Fecha OTD6	Estado-Observaciones
i20231C00304901	101498508	28/04/2023	2180B02 BOMBA TRANSFERE	471	000		02/05/2023		Pte entrada en almacén
i20231C00306901	101499387	02/05/2023	2180B02 BOMBA TRANSFER	889	000		02/05/2023		Pte entrada en almacén
i20231C00305901	101499385	02/05/2023	BOMBA TRANSFR COMBUSTI	827	000		02/05/2023		Pte entrada en almacén
i20231C00312901	101499410	02/05/2023	20031 ALTERNADOR 20KVA S	196689	000		02/05/2023		Pte entrada en almacén
i20231C00311901	101499399	02/05/2023	20031 ALTERNADOR 20KVA S	1124	000		02/05/2023		Pte entrada en almacén
i20231C00307901	101499388	02/05/2023	20031 ALTERNADOR 20KVA S	1180	000		02/05/2023		Pte entrada en almacén

Figura 1: Hoja excel ejemplo con equipos externalizados (fuente elaboración propia).

Esto mejora la situación, pero está desde luego muy lejos de lo necesario en muchas empresas. Es necesario otro grado de evolución más, y considerar la posibilidad de que el sistema de la empresa sea capaz de contactar automáticamente con el de terceros organizaciones y actualizar así la información. En otras palabras, es necesario mejorar la interoperabilidad del sistema de información de la empresa.

La interoperabilidad está definida por IEE (Institute of Electrical and Electronics Engineers) como "la capacidad de dos o más sistemas de intercambiar información y utilizarla" (traducción libre) (New World Enciylopedia, 2022). Como es lógico, es necesario normalizar entre organizaciones diversas cuestiones para que esta comunicación pueda llevarse a cabo. Puede tratarse desde distintos puntos de vista, como la interopabilidad organizativa, semántica o técnica, sin olvidar las infraestructuras y los servicios comunes que deben existir para que pueda llevarse a cabo a gran escala. Por ejemplo, en España la administración pública tiene definido un Esquema Nacional de Interoperabilidad (Portal administración electrónica , 2022)

Sin embargo, muchas veces las empresas no necesitan este grado de sofisticación y se limitan a interactuar utilizando tecnología lo más simple posible. Es frecuente que las organizaciones que ofrecen

servicios de outsourcing, provean una forma de acceder a la información desde la empresa que las contrata. Pueden exponer, por ejemplo una API (Interfaz para programación de aplicaciones), a la que los sistemas de información de la empresa pueden acudir a consultar cualquier información que se necesite en tiempo real y conocer el estado de la entidad externalizada. Por ejemplo, se puede conocer la localización física de un paquete que ha sido enviado a través de una compañia de transporte y conocer cuando llegará a su destino y anticipar la toma de decisiones antes de que el paquete acabe su viaje.

Esto sí es una gran avance, pero ¿qué ocurre si es necesario garantizar realmente que la entidad externalizado, tiene el estado declarado por la organización contratada para realizar el trabajo?. En otras palabras, ¿como garantizar la auditoria automática de los estados sucesivos que va alcanzando una entidad externalizada? Por ejemplo, ¿cómo garantizar el estado de conservación de una vacuna? Es un producto que a veces debe viajar desde el fabricante hasta el centro de salud manteniendo una temperatura determinada. ¿Realmente ocurre así?

En un somero análisis, una primera característica de cualquier sistema que opere en este exigente entorno es que la información debe ser inalterable por ambas partes. Y a la vez debe ser transparente, debe estar disponible para su consulta a todas las partes interseadas en el proceso. Debe ser tan confiable que pueda servir como base para que una auditoría se pueda realizar con todas las garantías legales. Así una vacuna que no ha respetado la temperatura de almacenamiento durante el transporte no puede ser inoculada a ninguna persona. El fabricante, el transportista y el centro de salud tienen que poder acceder a esta información para tomar decisiones.

Una solución posible es que este sistema esté basado en una red compartida que utilice tecnología blockchain. En este tipo de red se recurre a técnicas criptográficas para que la información sea inalterable, transparente, su acceso seguro, y la lógica que las gestiona puede incluso llegar a considerarse un contrato desde el punto de vista legal y llegar más allá del estricto control e implementar funciones de pago automático, por ejemplo.

Se trata en definitiva de interconectar el sistema informático interno a la empresa, que realiza la funciones de backoffice necesarias para externalizar un proceso, con un sistema externo a ella, que realiza el seguimiento de la entidad externalizada. En el ejemplo, todos los actores intervinientes en la producción, transporte e inoculación de la vacuna, deben poder acceder a los registros de temperatura de los sensores de las distintas neveras, para garantizar que ha mantenido la cadena de frío.

1.2. Blockchain.

Blockchain es un conjunto de tecnologías que permiten la transferencia de un activo sin intervención de terceros. Utiliza un registro de las transacciones y seguimiento de activos ocurridos en una red (IBM, 2022a).

Los activos pueden ser cualquier cosa que interese ser rastreada: propiedad intelectual, patentes, herramientas, dinero, mercancías, etc. El objetivo es reducir el riesgo de las transacciones y reducir los costes.

El registro de transacciones contiene la información de todas las transacciones realizadas y está replicado en todos los nodos que forman parte de la red blockchain. La autenticidad la verifica la red de nodos, y la transferencia de valor se realiza a través de un consenso (BBVA, 2022a). La información que contiene el registro es inalterable y solo los miembros autorizados de la red tienen acceso a ella.

La red puede hacer el seguimiento de cualquier activo, y dado que los usuarios comparten una fuente de datos única, pueden también ver los detalles de cualquier transacción, lo que produce confianza en las partes involucradas.

1.2.1. Registro de la blockchain.

Cada transacción que se produce en la red se registra como un bloque de datos. Estas transacciones representan el traspaso de un activo, un movimiento y pueden almacenar cualquier tipo de información, la temperatura de mercancías perecederas, quién la envió, su fecha de caducidad, dirección de envío, etc. (Pabex, 2021)

Estos bloques unidos forman una cadena a medida que el activo se va moviendo, se va transaccionando y forman una secuencia de transacciones que se unen de forma segura para evitar su alteración. Cada bloque registra cual es el bloque anterior y en caso de que una transacción contenga un error se añade una nueva transacción para que la información sea la correcta. Ambas serán visibles para toda la red.

Así cada bloque que se añade a la cadena refuerza la verificación de todos los bloques. Esto dificulta que cualquier atacante intente modificar algún bloque, bloque que por otro lado está replicado en otros nodos de la red. Tendría que modificarlo en todos los registros. La forma de registrar cada bloque y el siguiente es a través de su Hash.

Este hash es una función criptográfica a cuya entrada (la información del bloque, texto al fin y al cabo,) se le aplican una serie de operaciones obteniendo una salida, que también es texto. Esta operación además es irreversible, es decir, no se puede obtener el contenido del bloque a partir del hash obtenido por lo que se protege la privacidad de la información.

En la siguiente figura se puede observar que, ante cualquier modificación en un bloque, la cadena se rompe porque el hash que la une también se verá modificado.

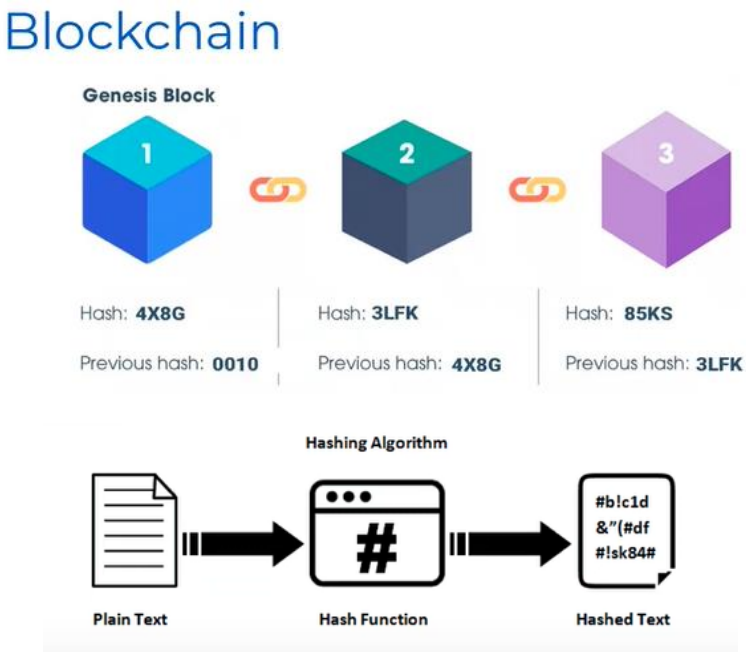

Figura 1: Cadena de Bloques (Pabex, 2021) .

1.2.2. Política de consenso.

Los nodos de la blockchain forman un tipo de red descentralizada, es decir, es una red sin jerarquía en la que ningún nodo puede organizar la forma de comportamiento del resto de nodos de la red. Cada nodo podría tomar decisiones individuales por lo que se establece una forma de política de consenso.

La política de consenso es la forma que tienen los nodos de la red para decidir cual es el próximo estado de la blockchain, es decir, cual es el siguiente bloque a añadir a la blockchain.

Generalmente se necesitan que el 51% de los nodos aprueben una transacción para que se añada a la blockchain.

Existen varios algoritmos utilizados para establecer la política de consenso:

- Proof of work (PoW).

  Primer algoritmo en utilizarse (lo usa bitcoin) y su robustez se basa en el alto coste para un atacante en convertirse en el 51% de los nodos de la red. Depende de que la red sea muy grande para ser seguro. Bitcoin o Ethereum lo utilizan.

- Proof of Stake (PoS).

  Es un algoritmo PoW al que se añade la mejora basada en que un nodo merece mayor confianza cuanta más moneda aporte. Así los nodos se configuran para que dispongan de una cierta cantidad de moneda. Así si un nodo realiza prácticas irregulares se ve penalizado por la red reduciendo la cantidad de moneda aportada. Este algoritmo surgió en la red Ethereum.

- Proof of Autorithy (PoA).

  Se utiliza en redes privadas en la todos los nodos pertenecen a organizaciones conocidas que han sido verificadas por aquellas que ya estaban en la red.

  Como la red no puede tener participantes anónimos no es necesario que se realice el proceso de minado ni que exista una moneda que recompense su trabajo.

1.2.3. Tipos de redes.

Pueden ser de 3 tipos: públicas, privadas y permisionadas o de consorcio (IBM, 2022a) (IBM, 2022b).

- Públicas.
  En este tipo de redes blockchain cualquiera puede unirse, pero utilizan gran potencia computacional, poca privacidad para las transacciones y la seguridad es más débil, los participantes se pueden mantener en el anonimato. Bitcoin es un ejemplo de red pública.
- Privadas.
  En este caso una sola organización controla quién puede unirse a la red, utilizan la identidad del participante para permitirle su acceso y privilegios, cuando ejecutar un algoritmo de consenso y mantiene el registro de transacciones. Aumenta la confianza de los participantes en función del caso de uso de que se trate. Suele permitirse la visualización sobre todas las transacciones.
- De consorcio o permisionadas.
  Este tipo de redes está constituido por varias organizaciones en las que algunos nodos elegidos son los que realizan el consenso. Se utiliza cuando los participantes deben estar autorizados y deben corresponsabilizarse de la red.

  Ejemplo de redes privadas o permisionadas se puede citar a Hyperledger.

1.2.4. Smart contracts.

Nick Szabo desarrolló el concepto en década de los 90 como un programa que tiene como objetivo ejecutar o documentar automáticamente acciones legalmente relevantes en función de los términos de un contrato.

Se introdujeron en la red Ethereum para añadir la lógica de negocio en las transacciones. En la actualidad el resto de redes también los han incorporado y permite programar cualquier lógica de negocio necesaria como consultar información o registrarla. El mecanismo es similar a un API: dada una entrada se le aplica una función para obtener una salida.

El lenguaje más usado en la actualidad es Solidity (su sintaxis es muy parecida a la de Javascript) pero también se utilizan otros como Vyper (basado en Python) o Java.

1.3. Mantenimiento en la Industria Aeronaútica.

El mantenimiento en general es una cuestión a considerar cuando se adquiere una inversión. Por ejemplo, los costes anuales del mantenimiento de una vivienda para mantenerla en el mismo estado en el que se compró pueden oscilar entre el uno y el cuatro por ciento de su valor original. No se puede automatizar y es intensivo en mano de obra especializada, así que es caro.

Por otro lado, la falta de mantenimiento también tiene un alto coste. Esto se puede observar especialmente en las catástrofes que se producen por falta de mantenimiento como el caso del puente que colapsó en Génova en el año 2018. En este caso no sólo es el coste de la pérdida del puente y la catástrofe asociada, sino que además tuvo que ser sustituido por uno nuevo (BBC News Mundo, 2020).

Para garantizar que sucesos como este no tengan lugar, el mantenimiento en la industria aeronaútica se realiza a través de un tipo de operaciones de mantenimiento llamadas inspecciones o verificaciones. En ellas se trata de controlar el estado de las distintas partes de la aeronave. Estas inspecciones varían en frecuencia, su duración y minuciosidad. Cada fabricante describe para un tipo de aeronave determinada, una combinación de horas de vuelo, despegues, o meses de tiempo y las inspecciones o tareas de mantenimiento a realizar para garantizar la operatividad de la aeronave en condiciones absolutas de seguridad.

Sirva como ejemplo una revisión de cierto tipo podría realizarse cada 40 horas de vuelo o cada mes. Lo que ocurra antes. Inspeccionar elementos importantes de la aeronave y realizar ciertas tareas de mantenimiento ligeras, como tomar muestras de aceite del motor para analizarlas en busca de partículas metálicas. Además, si por cualquier razón la aeronave despega del suelo, se contabiliza, y es posible que se deban inspeccionar los motores cada 120 despegues de forma independiente.

El trabajo para inspeccionar una aeronave varía en función de su minuciosidad y puede variar desde el orden de varios días a varios meses.

En función de la criticidad de cada componente de la aeronave para el vuelo se programan inspecciones para componentes internos. Así es posible cada 500 horas de vuelo tener que inspeccionar el control de combustible de un motor. Este tipo de inspecciones se debe realizar con el control de combustible desmontado, en un banco de prueba específico que realiza una serie de pruebas para garantizar el correcto funcionamiento del control de combustible.

Generalmente la inspección de los componentes de la aeronave es posible realizarla por la empresa propietaria hasta un cierto nivel, a partir del cual debe ser enviado a empresas especializadas. Es frecuente por tanto el envío de controles de combustible a un tercero. Esto naturalmente obliga a la empresa propietaria a disponer de un cierto stock de controles de combustible para no detener la inspección del motor y por agregación la revisión de la aeronave completa si es el caso. Los componentes también se averían y deben ser sustituidos.

Para conseguir ordenar todo este puzle de inspecciones que se entrecruzan en el tiempo y mantener una aeronave en condiciones de operatividad y seguridad, se realiza un trabajo de programación del mantenimiento de cada elemento susceptible de ser inspeccionado en algún momento. Y para esto es necesario garantizar la trazabilidad de los elementos inspeccionados y las tareas de mantenimiento realizadas en cada inspección.

1.3.1. Ciclo de mantenimiento genérico.

Cuando es necesario realizar un mantenimiento a algún componente de la aeronave de forma independiente, es habitual controlarlo administrativamente a través de alguna forma de orden de trabajo y a partir de ahí comenzar un ciclo estandarizado de mantenimiento. Este ciclo abarca desde la propia cumplimentación de la orden de trabajo con los datos del componente, selección del taller o empresa que la va a realizar, recursos previstos a consumir en su realización, traslados físicos del componente, registro de los recursos realmente consumidos y reintroducción del componente en la cadena de suministro en condiciones de ser utilizado.

Esto tiene un reflejo documental desde el punto de vista del control de la calidad que no es objeto del presente trabajo.

Este control administrativo tiene por objeto identificar en qué punto del ciclo de vida de la orden de trabajo se encuentra el componente, recursos dedicados a su mantenimiento, compromisos contractuales contraídos en caso de enviarlo a otra empresa a reparar, control de garantías, control presupuestario o registro de facturas, entre otros.

Este trabajo va a tratar únicamente el caso en el que un componente es enviado a mantener a una empresa externa a la organización.

1.3.2. Ciclo de vida del mantenimiento externalizado.

La particularidad del mantenimiento realizado en una empresa externa a la organización es que los recursos utilizados son financieros. Estos gastos se cargan a un presupuesto anual y deben ser contabilizados y controlados. Además, el componente debe transportarse físicamente desde las instalaciones de la organización que contrata el servicio, hasta la empresa externa que lo va a realizar y vuelta. Esto a veces significa enviarlo a otro continente y puede ser importante conocer su ubicación física en un instante determinado.

El ciclo de vida de una externalización típica con ligeras variaciones es:

- Identificación del componente a mantener.
- Identificación de la tarea de mantenimiento.
- Asignación de la orden de trabajo.
- Selección de la empresa externa que realizará el mantenimiento requerido.
- Identificación del coste de la operación y del contrato presupuestario a utilizar.
- Envío del material a la empresa que realizará la operación de mantenimiento.
- Recepción del material una vez mantenido.
- Regularización y registro del coste de la operación de mantenimiento.
- Facturación y pago.

1.3.3. Soporte informático del ciclo de mantenimiento.

En la actualidad, cada empresa mantiene un sistema informático para registrar las operaciones descritas en los anteriores epígrafes. Estos sistemas operan independientemente el uno del otro, de forma que no es posible conocer las vicisitudes del material una vez salen de la organización hacia la empresa externa.

Desde el punto de vista del control administrativo, los departamentos correspondientes deben ponerse en contacto por algún medio para conocer el estado del componente enviado a mantener. A veces, es necesario recabar información para decidir la major forma de solventar alguna necesidad sobrevenida mientras el equipo vuelve a la organización.

Teniendo en cuenta el punto de vista de la calidad, la empresa externa garantiza que su trabajo se realiza conforme a los expresado por el fabricante en sus manuales del componente, a través del cumplimiento de un certificado de calidad estandarizado en la industria, compatible con el de la organización que encarga el trabajo.

Esta es la razón de realizar una investigación para averiguar qué actividades son necesarias para desplegar una red blockchain en la actualidad, y añadir a este esquema un segundo sistema informático compartido por ambas organizaciones, que realice el seguimiento del componente enviado a reparar a partir de su salida de la organización hasta su vuelta una vez realizado el mantenimiento.

De esta forma no solo es posible conocer con absoluta transparencia el estado del componente externalizado en un momento dado, sino que el segundo sistema también puede ser utilizado como registro de las operaciones efectuadas desde el punto de vista de la calidad.

1.4. Herramientas.

Las herramientas utilizadas para el desarrollo del proyecto son las siguientes:

- Diseño: Thymeleaf, BootStrap y Pencil.
- Desarrollo: IntelliJ y Remix IDE.
- Base de datos y herramientas: MySQL y Workbench.
- Lenguajes: Java, Javascript y Solidity.
- Blockchain: Hyperledger Firefly, Hyperledger Fabric.

1.4.1. Diseño.

Para el diseño de la capa de presentación del cliente web se ha recurrido a dos herramientas que facilitan el control de la interacción con el usuario. Thymeleaf permite realizar la capa de presentación de las páginas web utilizando el lenguaje html y controlar la lógica de su comportamiento con tag específicos de Thymeleaf. En segundo lugar, Bootstrap permite controlar la estructura de la página, simplificar el control del comportamiento responsivo de las páginas web y utilizar un conjunto de elementos (iconos, menús, elementos desplegables, etc) comunes a todas las páginas de forma coherente y sencilla.

1.4.1.1. Thymeleaf.

Las páginas de la interfaz del cliente web se han realizado utilizando esta tecnología incluida en Spring Boot para manejar plantillas XML, HTML o HTML5.

En muchas ocasiones en las que se desarrolla en Java aplicaciones web se recurre a la tecnología jsp (java server page) para definir la presentación de las páginas web. En el caso de jsp es posible utilizando marcado directamente o a través de tags jsp. Pero el código jsp es engorroso hacer los cambios frecuentes que se necesitan en el desarrollo (thymeleaf, 2020), cualquier modificación necesita ser desplegada en el servidor para ver los cambios.

Sin embargo, en el caso de Thymeleaf, la lógica de la página está separada de la presentación. De esta forma es posible que el diseñador trabaje en local para realizar el diseño de la plantilla de la página utilizando html, y comprobando los cambios con cualquier navegador, que simplemente ignorará las etiquetas Thymeleaf. Sólo debe mantener las etiquetas para que al desplegar la página funcione correctamente.

Mientras, el programador trabaja la lógica de la presentación a través de etiquetas Thymeleaf, y esa misma plantilla junto con la presentación que el diseñador añade sea procesada en el motor de plantillas. Así es posible trabajar de forma conjunta diseñador y programador para el desarrollo de la capa de presentación de una aplicación web.

Thymeleaf tiene soporte para la internacionalización de contenido, un alto rendimiento al incorporar un sistema de caché y soporta OGNL (Object Graph Navigation Language) para la definición de expresiones y el acceso a las variables proporcionadas a la plantilla como entrada.

Por ejemplo, a través de la etiqueta th:text, visualiza un mensaje contenido en la variable msg:

```
<div th:if="${msg != null}" class='alert alert-success' th:text="${msg}" role='alert'></div>
```

Figura 2: Etiqueta th:text muestra una cadena en la variable msg (fuente: elaboración propia).

Se observa una estructura de repetición en la que se recibe el objeto listadoPeticiones enviado desde un servidor en el backend, y se itera con él accediendo a sus propiedades:

```
<tr th:each="peticion : ${listadoPeticiones}">
    <td th:text="${peticion.numeroPeticion}"></td>
    <td th:each="equipo : ${peticion.equipos}" th:text="${equipo.referencia}"></td>
    <td th:each="equipo : ${peticion.equipos}" th:text="${equipo.nombre}"></td>
    <td th:each="equipo : ${peticion.equipos}" th:text="${equipo.numeroSerie}"></td>
    <td>
        <a th:href="@{/cpeticiones/editar/{id} (id=${peticion.idPeticion}) }"
           th:text="'editar'" class="btn btn-primary btn-sm" role="button" title="Editar el
       curso."><i class="fa fa-pencil" aria-hidden="true"></i></a>

        <a th:href="@{/cpeticiones/borrar/{id} (id=${peticion.idPeticion}) }"
           th:text="'borrar'" onclick="return confirm('¿Estas seguro?')" class="btn btn-danger
       btn-sm" role="button" title="Eliminar la petición.">
            <i class="fa fa-trash" ariahidden="true"></i></a>
    </td>
</tr>
```

Figura 3: Estructura de repetición con etiqueta th:each (fuente: elaboración propia).

Posibilitando la modificación y representación dinámica de datos obtenidos a través de una aplicación que se está ejecutando en un servidor backend.

Se escoge esta tecnología por su flexibilidad, facilitando el trabajo en equipo e integrarse perfectamente con las tecnologías escogidas para el desarrollo del backend: Java/Spring Boot.

1.4.1.2. Bootstrap.

Es un potente toolkit para desarrollo front-end que facilita construir sitios web completamente responsivos, contiene un sistema de grid (Bootstrap, 2022a) para el desarrollo de las páginas web y ofrece potentes plugins javascript (botones, menús de navegación, iconos, etc.) para dotar los sitios construidos de interactividad con el usuario (BootStrap, 2022b).

Figura 4: Diferentes estructuras para un grid de Bootstrap (Bootstrap, 2022a).

Diseñada y creada por un diseñador y un desarrollador de Twitter (Mark Otto y Jacob Thornton), a mediados del 2010 recibió el nombre de Blueprint (BootStrap, 2022c). Al principio fue un desarrollo utilizado únicamente de forma interna dentro de la propia Twitter hasta que fue liberada en Agosto de 2011. Desde entonces lleva alrededor de treinta releases (BootStrap, 2022d), y ha sido reescrita tres veces. Bootstrap 3 se lanzó para hacerla responsiva e incluir el por defecto el desarrollo de aplicaciones "first mobile" (realizar en primer lugar la aplicación móvil). Continúan añadiéndole nuevas funcionalidades y se encuentra en la versión 5 que abandona jQuery (jQuery, 2022)para sustituirlo por VanillaJS (VanillaJS, 2022), deja de soportar Internet Explorer 10 y 11 y los elementos son visualizados de la misma forma en todos los navegadores soportados, entre otras características nuevas (Bla, 2021).

En este trabajo se ha utilizado el sistema de grid, agrupado la información en tarjetas (card), aplicado estilos o definido barras de navegación:

```
<div class="row">
    <div class="col-12">
        <div class="card">
            <div class="card-header bg-info"> Datos principales </div>
            <div class="card-body">
                <div class="row">
                    <div class="col-4 required">
                        <label for="numeropeticion" class="form-label">Número de petición</label>

                        <input type="text" class="form-control" th:field="*{numeroPeticion}" id="numeropeticion"
                               name="numeropeticion"
                               placeholder="Escriba el número de la petición" required="required">
                    </div>
                    <div class="col-4 required" th:each="equipo: ${peticion.equipos}">
                        <dl>
                        <dt >Referencia</dt>
                        <dd th:text="${equipo.referencia}" ></dd>
                        </dl>
                    </div>
                    <div class="col-4 required" th:each="equipo: ${peticion.equipos}">
                        <dl>
                            <dt >Nombre</dt>
                            <dd th:text="${equipo.nombre}" ></dd>
                        </dl>
                    </div>
```

Figura 5: Estructura del grid de página 12 columnas. Primera fila 3 card de 4 columnas cada una (fuente: elaboración propia.)

En este caso se puede observar la división en doce filas y 4 columnas para mostrar distinta información agrupada en cards. También se aprecia el uso de tag Thymeleaf para referirse a variables cuyo contenido es pasado a la plantilla al procesarla por el motor de plantillas.

1.4.1.3. Pencil.

Pencil Project (Evolus, 2023) es una herramienta gráfica open-source, para el prototipado y diseño de mockups especialmente dirigida al diseño de interfaces de usuario. Puede ser instalada tanto en Linux, Windows como macOS y dispone de imágenes de los elementos gráficos habitualmente utilizados en aplicaciones móviles (Android o iOS), de escritorio o páginas web, con capacidad para enlazar entre sí todas las pantallas diseñadas. También es posible realizar diagramas de flujo, UML (Unified Model Language) o utilizarlo como visualizador de clipart de OpenClipart.org para poder añadirlos a los diseños.

Es la herramienta utilizada para realizar los mockups de las páginas web del proyecto por su simplicidad de uso y utilizar una interfaz clara y minimalista. En la parte izquierda se selecionan los mockups de los controles a utilizar, y arrastrándolos a la zona central de la pantalla se va contruyendo la interfaz.

Cada página o elemento pueden estar enlazados con elementos de otras páginas de forma que es posible simular el comportamiento esperado de la interfaz sin realizar ningún tipo de programación. Posteriormente, es posible reunirse con el cliente y experimentar el uso de la interfaz, para proporcionar feedback a los desarrolladores con sus observaciones.

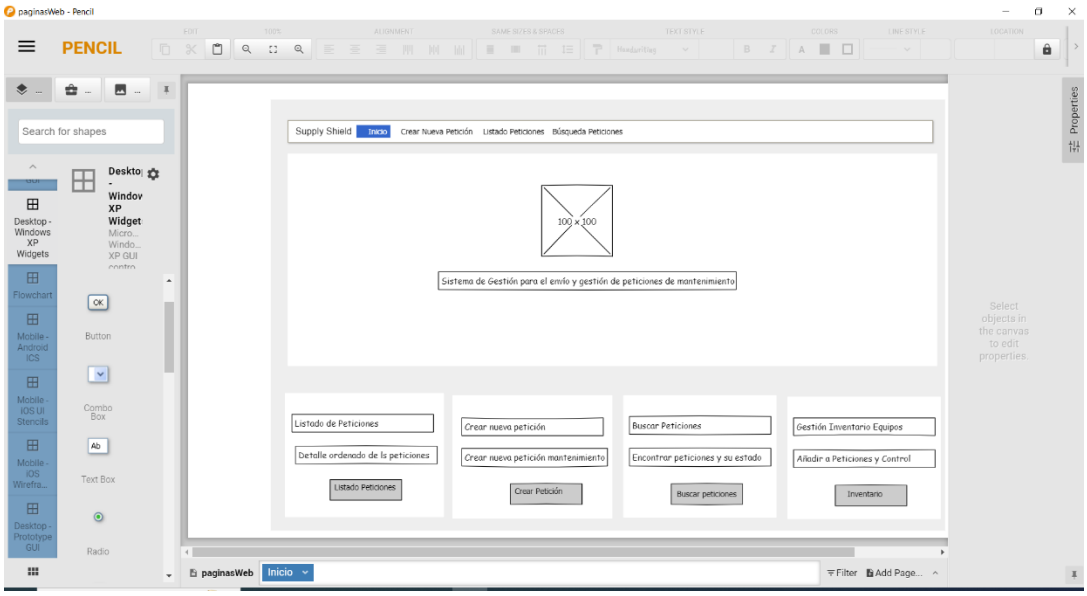

Figura 6: Interfaz Pencil. A la izquierda panel con controles (fuente: elaboración propia).

1.4.2. Entorno de desarrollo.

Este proyecto utiliza una diversidad de lenguajes: Java, Javascript, HTML, CSS, Thymeleaf y Solidity. Además, debe integrarse con una base datos y poder acceder a distintos servicios REST con sencillez. Se decidió utilizar IntelliJ por la enorme cantidad de soporte de lenguajes y funcionalidades que ofrece. Sin embargo, el entorno Remix IDE, especializado en el desarrollo de aplicaciones distribuidas para redes blockchain con Solidity, ofrece una serie de funcionalidades extra que han hecho utlizarlo en lugar de IntelliJ. Remix IDE es un entorno de desarrolloconcontiene soporte para cualquier versión del compilador y lenguaje Solidity, despliega los smart contract en una red de prueba y genera un api de forma automática para probar su funcionamiento a través de un sencillo formulario. Dispone además de una versión on-line muy completa que se utiliza desde el navegador.

1.4.2.1. IntelliJ.

El entorno de desarrollo escogido ha sido IntelliJ. Es una herramienta con un IDE relativamente ligero e intuitivo, que facilita la labor del usuario proporcionando una gran cantidad de funcionalidades para la inspección y depuración del código, integración de herramientas para el control de versiones, soportando una gran variedad de lenguajes, siempre con la misma interfaz de usuario.

IntelliJ publicó la primera versión, la IDEA, en 2001. Fue uno de los primeros IDE para lenguaje java disponibles con capacidad de refactorización de código, exploración avanzada de código e integra Junit. (Jetbrains, 2020).

En 2005 integra soporte para desarrollo web añadiendo HTMS, CSS y Javascript. Durante 2006 añade el soporte para internacionalización, para a partir del año 2007 soportar Spring, Hibernate, servicios web, Maven y Ruby. A partir del 2012 Kotlin se incluye en el IDE para el desarrollo para plataformas de móvil

y en 2017 una nueva herramienta: cliente http integrado con el editor. La herramienta de control de versiones se incluye a partir de 2018, consiguiendo a partir del 2020 añadir la revisión y merge de las solicitudes de cambios de Github.

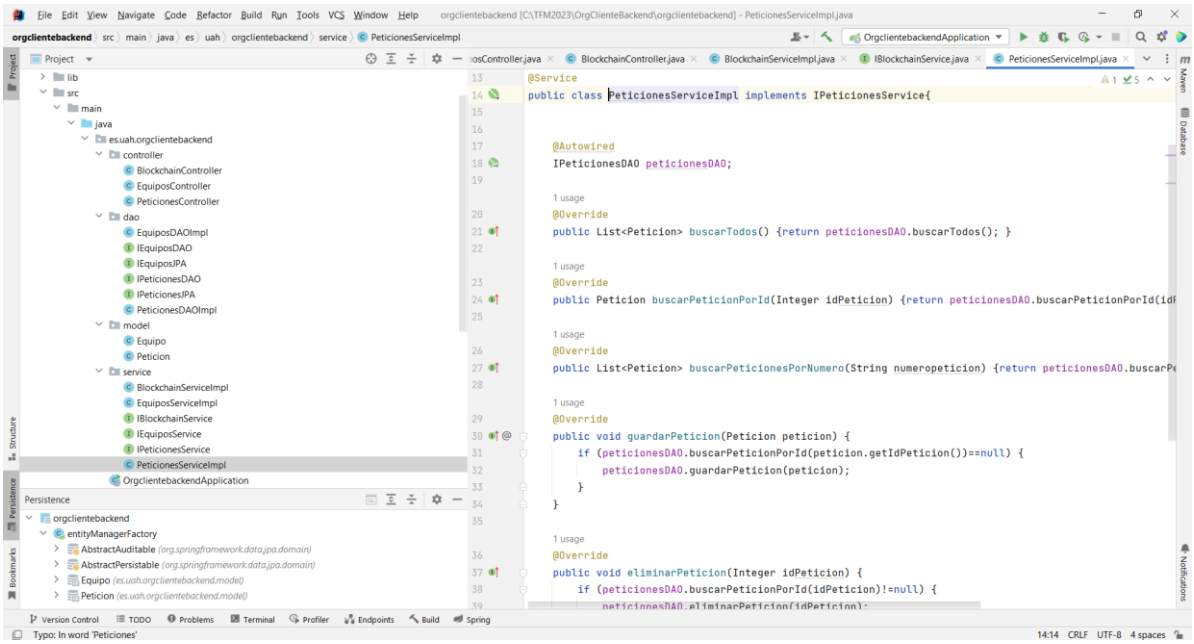

Figura 6: Interfaz IntelliJ (fuente elaboración propia).

En este rápido repaso de la historia de este entorno, se observa la gran cantidad de funcionalidades añadidas en todos estos años. Una gran herramienta para el desarrollo en muchos de los lenguajes existentes, compartiendo una misma interfaz gráfica. Esto ha hecho elegirla frente a otras herramientas, también de una gran calidad, pero no tan potentes en la ayuda al usuario, como Eclipse o Netbeans. Aunque existe una versión Comunity Edition gratuita, para este trabajo se ha utilizado la versión Ultimate, más completa, que se ofrece con una licencia sin coste para todos los estudiantes.

1.4.2.2. Remix IDE.

Para el desarrollo del smart contract de la red blockchain se ha recurrido a utilizar Remix IDE. IntelliJ dispone de un plug-in para Solidity, pero Remix ofrece además compilación, pruebas y despliegue en una red de prueba.

Aunque se ha utilizado la versión on line accesible a través de la dirección https://remix.ethereum.org/ , también dispone de una versión de escritorio y un plugin para VS Code.

En la siguiente figura, se puede observar el aspecto de su interfaz. En la ventana del editor apararece el código del contrato de este proyecto. En el apartado izquierdo figura por un lado el resultado del autocompilado, marcado con un círculo verde y un check, junto al botón Deploy para desplegar el contrato en la red de prueba (o la que se haya configurado anteriormente). Más abajo se sitúa la dirección del contrato y el formulario que permite probar las funciones programadas introduciendo distintos valores para los parámetros. En naranja las funciones de creación o actualización y en azul las funciones de lectura.

En la parte inferior, debajo del editor de código, dispone de una consola donde se muestran las operaciones que se van realizando en la red de prueba y los valores registrados.

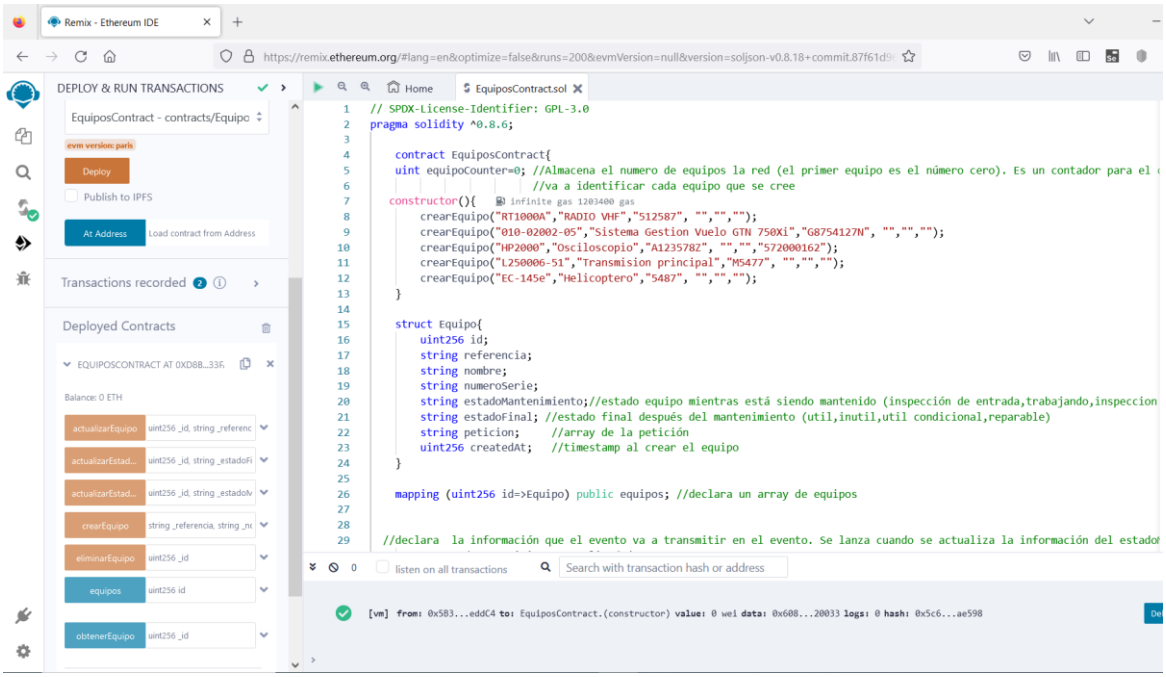

Figura 7: Interfaz Remix IDE (fuente: elaboración propia).

1.4.3. Base de datos y herramientas.

Para almacenar la información de la organización se ha utilizado la base de datos MySQL. Es una base de datos relacional que almacena la información en tablas relacionadas entre sí. Se busca poder almacenar datos con estructuras que faciliten la gestión de las relaciones que mantienen entre ellas.

Para facilitar la gestión de la base de datos, se utiliza el interfaz gráfico que la propia distribución contiene. Con un bajo consumo de recursos permite realizar consultas, crear bases de datos, crear tablas, añadir información, eliminarla o modificarla. También se ha utilizado para ejecutar scritps sql e insertar datos de prueba o crear tablas. Son operaciones sencillas pero suficiente para lo necesario en este proyecto.

Para realizar pruebas de los apis del backend para el acceso a la base de datos y del que Hyperledger FireFly genera para interactuar con la red blockchain se ha utilizado la herramienta Postman.

1.4.3.1. MySQL Database Server.

MySQL es una base de datos relacional con doble licencia, existiendo una versión open source (Comunity Edition) (Oracle, 2023a) y otra comercial (Oracle, 2023b). Inicialmente desarrollada por la empresa MySQL AB, fue adquirida en 2008 por la emblemática Sun Microsystem, que a su vez fue comprada por Oracle en el año 2010. La estructura de la base de datos y la organización física de los ficheros busca optimizar la velocidad de proceso. Su modelo lógico compuesto de objetos como las bases de datos, tablas o vistas permite una gran flexibilidad de uso. Es posible generar índices de búsqueda, distintas formas de relaciones entre las tablas, o programar procedimientos almacenados. Utiliza SQL como lenguaje de acceso a la base de datos.

Se puede instalar en una variedad de entornos como en ordenadores de escritorio o portátiles, servidores de aplicaciones, servidores web o incluso clusters de máquinas conectadas en red. También se ofrece en forma de librería para instalarla en sistemas embebidos. Tradicionalmente muy usada en servidores de internet, utilizando como servidor web Apache bajo sistema operativo Linux (LAMP – Linux, Apache, MySQL y php, Python o perl) o windows (WAMP).

1.4.3.2. Herramienta MySQL Workbench.

Se trata de una herramienta gráfica para administrar bases de datos MySQL (Oracle, 2023c). Facilita la creación y gestión de conexiones a la base de datos, ejecutar consultas SQL, genera modelos gráficos del modelo de bases de datos existente, crea y edita tablas, columnas, disparadores, índices, vistas y particiones de la base de datos. Tambien tiene capacidad para administrar usuarios, realizar copias de seguridad, monitorizar el rendimiento de la base de datos y realizar migraciones desde otros sistemas de gestión de bases de datos como Microsoft SQLServer, PostgreSQL o SQLite.

Se ha utilizado esta herramienta debido a su potencia, flexibilidad, reducido consumo de recursos simplicidad y sencillez de uso. Únicamente se ha utilizado para la ejecución de scripts para creación de la base de datos, inserción de datos de prueba, visualización de datos en pruebas del cliente del proyecto y generación de modelos E-R.

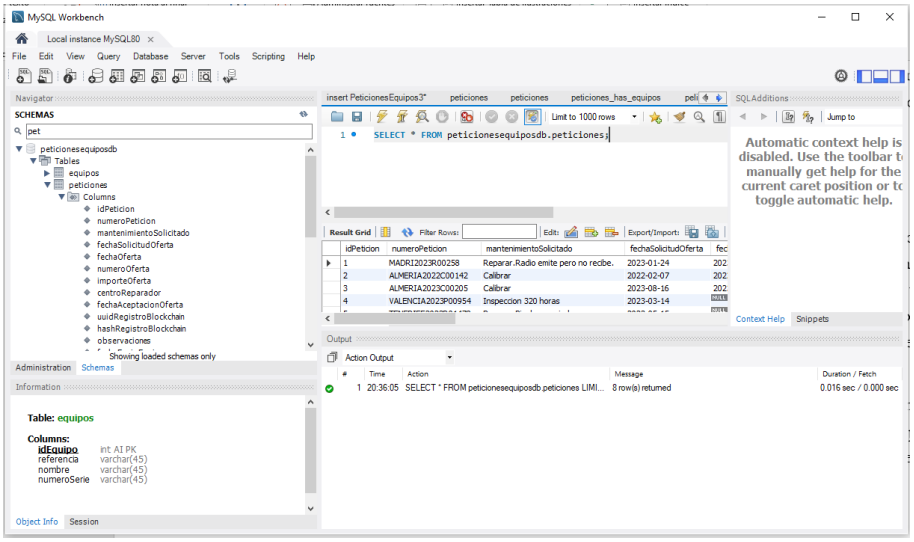

Figura 7: Interfaz MySQL Workbench (fuente: elaboración propia).

1.4.3.3. Postman.

Postman (Postman, 2023) es una herramienta que facilita la construcción y el uso de APIs a lo largo de su ciclo de vida. Dispone de un repositorio para almacenar, catalogar y ponerlas todos los artefactos creados en una plataforma común de forma que pueda se compartida por todo el equipo de desarrollo. Documentación, casos de test, flujos de trabajo, por ejemplo. Integra una serie de herramientas para facilitar el diseño, testeo, documentación y mocking de APIs. Para organizar las API en desarrollo utiliza workspaces personales, del equipo, cliente y públicos. Es una plataforma extensible que permite utilizar

herramientas de terceros para, por ejemplo, integrar Postman en plataformas de desarrollo continuo como GitLab.

En este proyecto se ha utilizado para realizar peticiones get, put, post o delete a las APIs de los microservicios y probar su funcionamiento. Se puede observar su interfaz la figura 96.

1.4.4. Lenguajes.

Para la implementación de este proyecto se ha utilizado Java para en las distintas capas del backend y el acceso a datos. La interacción de usuario en el frontend se ha programado utilizando el motor de plantillasThymeleaf y Javascript.

En el caso de la red blockchain se ha utilizado Solidity, que es lenguaje utilizado en redes ethereum, para programar un smart contract que gestione la información que la red almacena. No es necesario ningún lenguaje adicional para el desarrollo de la API REST, a través de la cual interactúa el cliente o el servidor con la red, porque Hyperledger FireFly la genera automáticamente a partir del smart contract.

1.4.4.1. Java.

Es un lenguaje de programación, creado por un equipo dirigido por James Gosling, dentro de la empresa Sun Microsystem. Comenzaron en 1991 con el objetivo de crear un nuevo lenguaje para que los dispositivos electrónicos de consumo pudieran comunicarse entre sí. Sin embargo, el equipo cambió el enfoque y cuando fue liberado en 1995 dotó a la incipiente World Wide Web de interactividad y elementos multimedia con los que enriquecerla. Utiliza una máquina virtual que interpreta un código intermedio (bytecode), al que se convierte el programa original, y es la máquina virtual la que traduce el bytecode para la computadora host sobre la que se está ejecutando. De esta manera el mismo programa puede ser ejecutado en multitud de máquinas virtuales JRE, Java Runtime Environment. A finales de la década de los 90 Java excedió los límites de internet y comenzó a ser utilizados en otros dispositivos: teléfonos móviles, computadoras domésticas y hasta robots exploradores de la NASA.

Dada la aceptación del lenguaje, se desarrollaron versiones del lenguaje específicas para computadoras personales (JSE-Java Stantdard Edition), JavaME para dispositivos integrados y Java Enterprise Edition para servidores.

Oracle Corporation se hizo cargo de la gestión de Java al adquirir Sun Microsystem en 2010 (Encyclopaedia Brytannica, 2022).

1.4.4.2. Javascript.

Lenguaje de programación que se puede utilizar en cualquier documento html con el objetivo dotar a los sitios web de interactividad dinámica con el usuario. Fue inventado en 1995 por Brendan Eich, cofundador del proyecto Mozilla, Mozilla Foundation y Mozilla Corporation. (Mozilla Contributors, Que es Javascript, 2023a). Es un lenguaje interpretado, orientado a objetos, basado en prototipos, débilmente tipado y dinámico. Aunque también es posible utilizarlo en otros entornos como (Mozilla Contributors,

JavaScript, 2023b). Aunque también es posible utilizarlo en otros como Node.js (entorno de ejecución multiplataforma para el lado del servidor) o Apache CouchDB (base de datos) , en este proyecto se usa únicamente en el lado del cliente: el navegador web.

El lenguaje está estandarizado (ECMAScript-262) y en la actualidad existen tres motores diferentes que interpretan el código Javascript: Chromium en el navegador Chrome, Webkit utilizado por el navegador Safari y SpiderMonkey, mantenido por la fundación Mozilla y primer intérprete desarrollado por Brendan Eich. En la actualidad lo utiliza Firefox.

Además, sobre el núcleo del lenguaje se han desarrollado api's, integradas en los navegadores, que permiten añadir aún más funcionalidad, como por ejemplo (Mozilla Contributors, What is Javascript?, 2023c):

- DOM (Document Object Model) permite manipular HTML y CSS de la página de forma dinámica en tiempo real. Aplicar estilos, ventanas emergentes o mostrar nuevo contenido son algunas de sus funcionalidades.
- Canvas y WebGL permiten crear animaciones 2D y 3D.
- Api's de audio y video como HTMLMediaElement capacitan al navegador para mostrar audio y video o utilizar la cámara del computador para realizar transmisiones.
- API's de Geolocalización, como Google Maps ofrecen funcionalidades como mostrar mapas, trazar rutas o geolocalizar al usuario.

1.4.4.3. Solidity.

Es un lenguaje desarrollado desde 2014 por colaboradores del proyecto Ethereum, específicamente diseñado para desarrollar y desplegar smartcontract sobre una red blockchain de Ethereum. A través de los smartcontracts se puede también definir y controlar el comportamiento de aplicaciones descentralizadas que se ejecutan sobre una red blockchian Ethereum (Solidity Team, 2023). El concepto smartcontract o contrato inteligente fue definido por Nick Szabo en el año 1994, antes de que existiera la tecnología para su implementación (Ethereum Team, 2023).

Su sintaxis es similar a la de otros lenguajes orientados a objetos, su tipado es estático y soporta herencia, tipos definidos por el usuario y el uso de otras librerías.

Se puede utilizar una gran variedad de editores para su uso desde el uso de plug-in para Visual Studio o IntelliJ hasta IDE como Remix, Etherum Studio o Hardhat (Ethereum, 2023).

1.4.5. Blockchain.

Para el despliegue de una red blockchain y la interacción de una aplicación web con ella se ha experimentado con dos posibilidades. La primera de ellas, Hyperledger Firefly es capaz de desplegar la red y generar un API Rest de forma automática, ocultando al desarrollador la complejidad de toda su estructura y los detalles de implementación de la API. Es la utilizada para el desarrollo del proyecto.

En segundo lugar, se presenta Hyperledger Fabric, que precisa que el desarrollador controle la estructura de la red blockchain desplegada y la implementación de la API Rest necesaria para interactuar con ella. Si el caso de uso precisa de este control, esta opción lo permite, pero precisa de un esfuerzo muy importante por parte del desarrollador que excede el alcance de este proyecto.

Ambas opciones se explican con mayor profundidad en el apartado 1.8.3.

1.4.5.1. Hyperledger FireFly.

Se trata de un stack completo de herramientas que permiten construir aplicaciones web para redes blockchain de forma rápida. Facilita la gestión de activos digitales, flujos de datos o transacciones en una red blockchain. Es un software open source, contribuido por la empresa Kaleido en el año 2021, que estandariza la forma de construir una red blockchain con varios miembros y las aplicaciones que la utilizan. Para ello comunica las aplicaciones de los miembros con la red a través de un API REST que se puede consumir de la forma tradicional desde las aplicaciones de los miembros de la red. En la actualidad es un proyecto de Hyperledger Foundation. (Hyperledger, 2021). Encapsula la complejidad existente entre la red blockchain y una aplicación empresarial. De esta forma los desarrolladores sólo se ocupan de la lógica del negocio.

Para poder realizar su trabajo de orquestación ofrece un gran conjunto de funcionalidades (Hyperledger, 2023a) observables en la siguiente figura:

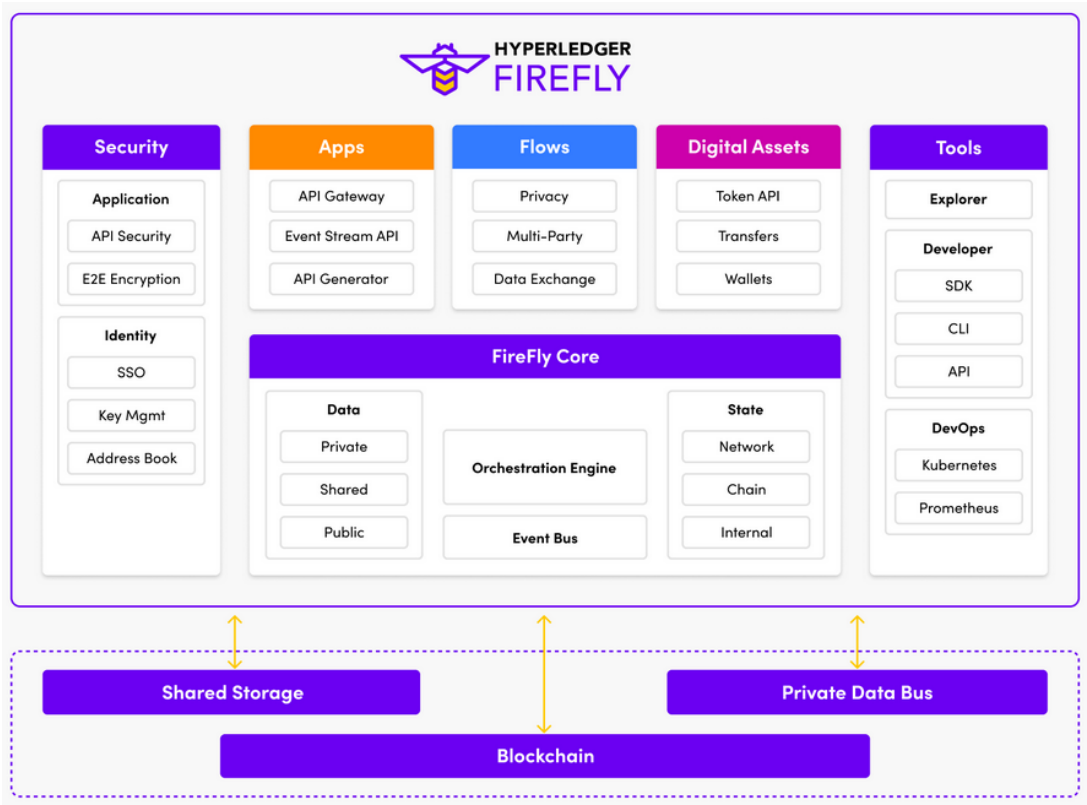

Figura 8: Hyperledger FireFly features, fuente (Hyperledger, 2023a)

1.4.5.2. Hyperledger Fabric.

Es una plataforma de código abierto modular y configurable que permite desplegar redes basadas en tecnología blockchain y contratos inteligentes creados en lenguajes de propósito general como Java, Go y Node.js.

La plataforma es permisionada, por lo que los participantes de la red se conocen, y por lo tanto el gobierno de la red se basa en la confianza Los protocolos de consenso pueden ser operados por una autoridad confiable, para que el consenso absoluto no sea necesario y no lastre el rendimiento. Además, aprovecha protocolos de consenso que no requieren de una criptomoneda para impulsar la ejecución de smart contracts, lo que contribuye a reducir posibles ataques al sistema.

En conjunto, todas estas características hacen de esta plataforma una de las de mejor rendimiento, tanto desde el punto de vista de las transacciones como de la latencia para confirmarlas (Androulaki, 2020).

DESARROLLO DEL PROYECTO

1.5. Modelo de Proceso de Software.

Para el desarrollo de cualquier proyecto la Ingeniería de Software propone cinco actividades estructurales (Pressman, 2.5.2 Fases del proceso unificado, 2010b):

- Comunicación.
  En contacto con el cliente, perfila a los participantes del proyecto para entender sus objetivos y reunir una serie de requerimientos que definan las características y funciones del software a desarrollar.
- Planeamiento.
  De esta actividad se obtiene un plan del proyecto que describa las tareas técnicas a realizar, los riesgos probables, los recursos necesarios, productos obtenidos y una programación de las actividades a realizar.
- Modelado.
  Se crean modelos de los objetos del proyecto, las relaciones que los unen o las funciones que realizan, más o menos refinados, con el objetivo de entender major los requerimientos del proyecto y el diseño que los cumple.
- Construcción.
  En esta actividad se genera el código del proyecto y las pruebas necesarias para descubrir errores que se haya podido incurrir.
- Despliegue.
  El software producido se entrega al cliente para que lo utilice y le proporcione al equipo de desarrollo realimentación de su uso.

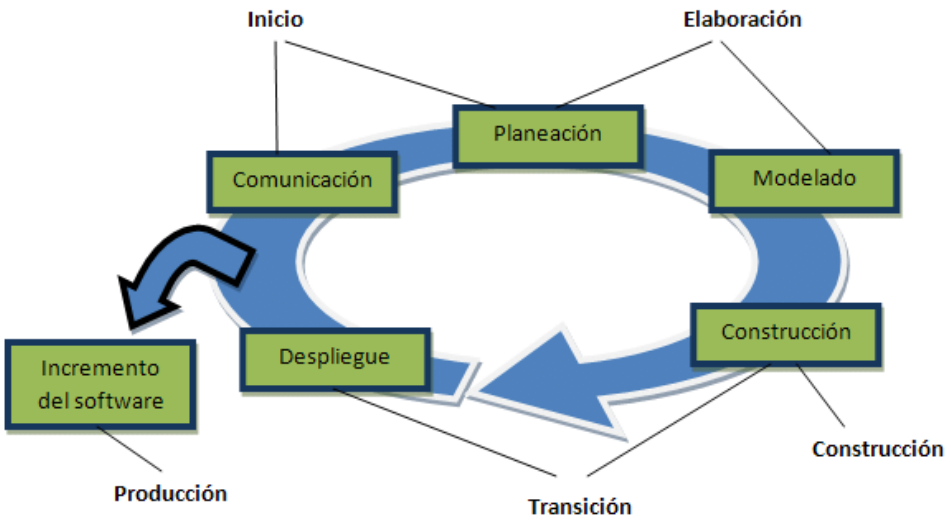

Figura9: Actividades Proceso Software General vs Proceso Unificado, fuente (Pressman, 2010b).

En este proyecto en concreto, se ha optado por utilizar para su desarrollo el modelo de proceso de software Proceso Unificado. Este modelo esta descrito en el libro Unified Process (Jacobson, 1999) por Ivar Jacobson, Grady Booch y James Rumbaugh que lo describen como "impulsado por el caso de uso, centrado en la arquitectura, iterativo e incremental". Los autores pretenden con este modelo reunir las mejores características de los modelos tradicionales del proceso de software junto a los principios del desarrollo ágil (Beck, 2001). Este proceso otorga gran importancia a la arquitectura y "ayuda a que el arquitecto se centre en las metas correctas, como que sea comprensible, permita cambios futuros y la reutilización" (Jacobson, 1999). Además su flujo de trabajo es iterativo e incremental (Pressman, 2.5.2 Fases del proceso unificado, 2010b).

Este modelo organiza las cinco actividades estructurales anteriormente descritas en cinco fases:

- Fase de inicio o concepción.
  En esta fase se realizan actividades de comunicación con el cliente y de planeamiento del proyecto. En colaboración con los interesados, se describen los requisitos del negocio mediante un conjunto de casos de uso, una arquitectura del futuro sistema que permita satisfacerlos y un plan que, de forma iterativa e incremental, implemente el proyecto en una serie de entregas de software.
- Fase de elaboración.
  En esta fase se mejoran y amplian los casos de uso desarrollados en la fase anterior y se realiza el modelado de la arquitectura a través de los modelos de casos de uso, de los requerimientos, del diseño, de la implementación y el despliegue. Además se revisa el plan del proyecto para asegurar que el alcance, los riesgos y la fecha de entrega son posibles o precisan ajustes. Proporciona una arquitectura viable aunque no todas las funciones que el sistema software debe ofrecer.
- Fase de Construcción.
  Se realiza la actividad de construcción del software desarrollando los componentes necesarios o utilizando los desrrollados por terceros. El objetivo, es implementar una solución para cada caso de uso descrito por los usuarios del sistema. Para ello, se refinan los modelos de requisitos y diseño que se van a implementar en el incremento de software para, finalmente, realizar el código fuente y realizar pruebas unitarias, de componentes e integración. Para las pruebas de aceptación se utilizan los casos de uso.
- Fase de transición.
  Esta fase incluye el final de la actividad genérica de construcción y las primera etapas de la actividad general despliegue. El objetivo es entregar el software en fase beta al usuario para que informe de errores e información acerca de cambios que considera positivos. Si es necesario se genera la documentación de apoyo necesaria para poner el incremento de software en producción.
- Fase de producción.
  Es la actividad general de despliegue. Se observa el uso que el usuario hace del software, la infraestrucra sobre la que se opera y se informa de los defectos encontrados o cambios que se propongan para el siguientes incremento.

Todas estas fases no ocurren de forma secuencial, sino escalonada, y no todas las tareas se utilizan en todos los proyectos, sino que el equipo de desarrollo adapta el Proceso Unificado a sus necesidades. Es un buen punto de partida para que, ingenieros de software junior al aplicar metodologías ágiles, dispongan de una guía de referencia (Pressman, 2010d). Esto permite apoyarse en el conocimiento de ingenieros más experimentados y reflexionar acerca de la necesidad o no de las tareas del proceso unificado a incluir en un proyecto. "El Proceso Unificado Agil adopta una filosofía en serie para lo grande e iterativa para lo pequeño".

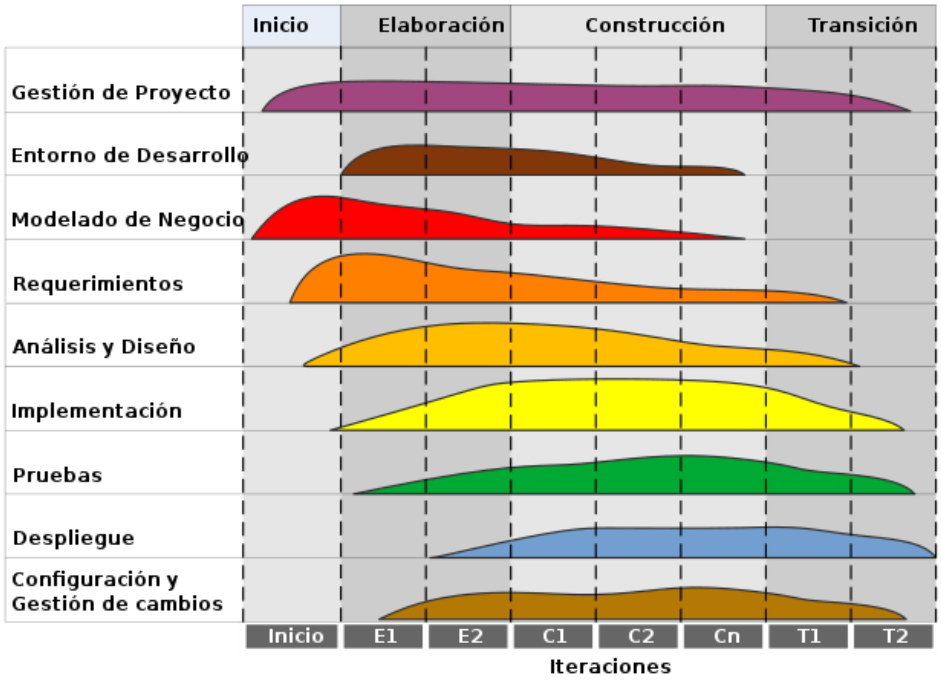

Figura 10: Proceso Unificado. Escalonamiento de las tareas. (Lantolin, 2020)

1.6. Fase de Inicio o concepción.

"En esta fase se establece el entendimiento básico del problema, las personas que quieren una solución, la naturaleza de la solución, así como la eficacia de la comunicación y colaboración preliminares en los otros participantes y el equipo de software." (Pressman, 2010c)

El objetivo del proyecto, como prueba de concepto, es desarrollar un software que permita gestionar el mantenimiento de un componente aeronáutico externalizado a otra empresa y evaluar la forma de realizar el seguimiento durante el tiempo que dure la externalización en el futuro.

El esquema de funcionamiento está reflejado en el siguiente caso de uso general:

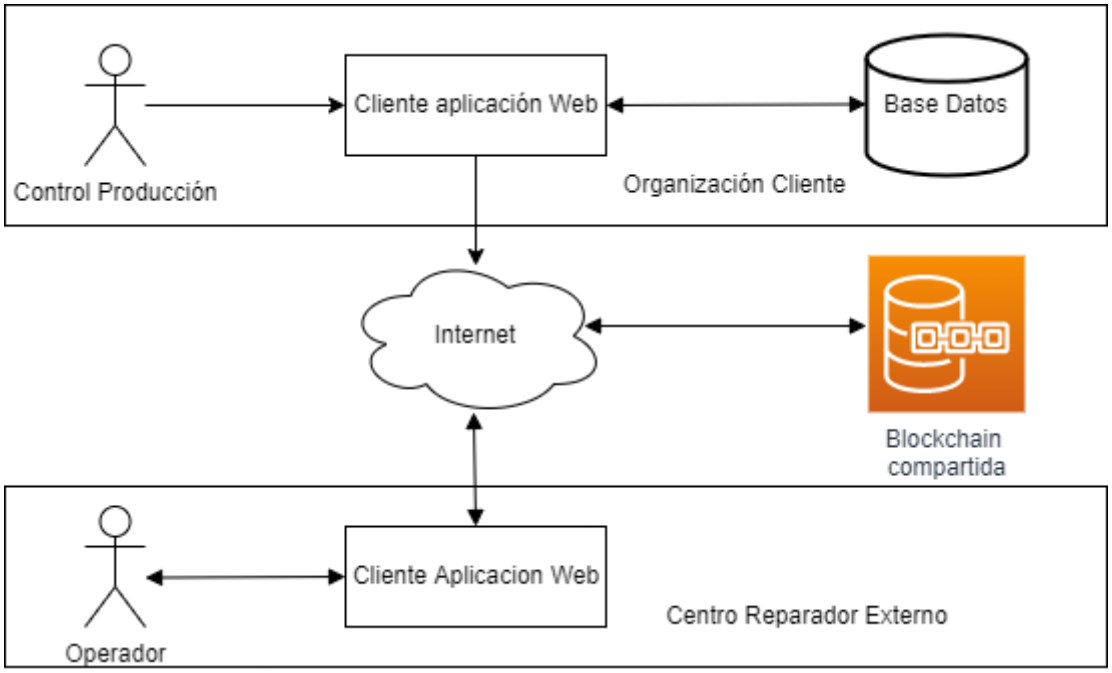

Figura 11: Esquema general del caso uso desarrollado en el proyecto, (fuente elaboración propia).

El usuario Control de Producción registra una petición de mantenimiento a un Centro Reparador Externo. Para ello utiliza el interfaz web de la aplicación, y negocia los términos contractuales de la misma utilizando métodos de comunicación tradicionales (correo electrónico, teléfono, contrato). Los términos negociados junto a los datos de identificación del equipo se utilizan para iniciar una transacción en la blockchain compartida.

Una vez el equipo a mantener abandona físicamente la Organización Cliente, el operador de la empresa Centro Reparador Externo registra sus vicisitudes en la Blockchain compartida a través de su interfaz web.

Así el operador puede registrar cuando se recogió el equipo, si se ha realizado la tarea de mantenimiento contratada, mecánico que la realiza, herramientas, fecha estimada de devolución a la Organización Cliente o fecha de entrega real.

Una vez el equipo vuelve a las instalaciones de la Organización Cliente, el usuario Control de Producción cerrará la petición de mantenimiento registrándolo a través de la interfaz web.

De esta forma quedan registrados en la blockchain tanto los términos del contrato como las vicisitudes del equipo y pueden ser consultados en cualquier momento por ambos usuarios.

Este sistema operará de forma independiente al resto de sistemas software de las organizaciones que lo utilizan. Cada usuario dispondrá de un ordenador con sistema operativo Windows con conexión a internet, lo menos potente posible, en el que realizar sus funciones.

1.6.1. Casos de uso preliminares. Organización Cliente.

Según la descripción realizada en el apartado anterior se pueden inferir dos actores y varios casos de uso para registrar las peticiones:

- Añadir una nueva petición de mantenimiento.

- Buscar una petición de mantenimiento por referencia y número de serie.
- Editar una petición de mantenimiento para actualizarla.
- Borrar una petición de mantenimiento.

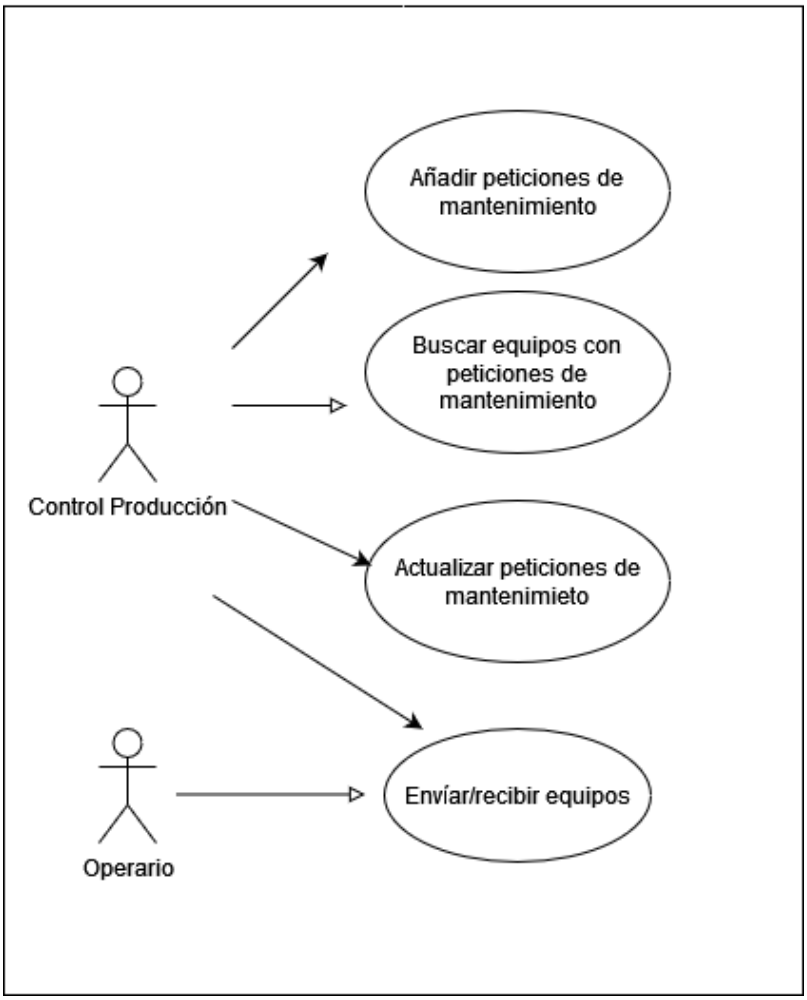

Figura 12: Diagrama de casos de uso, (fuente elaboración propia).

El futuro sistema de trazabilidad basado en tecnología blockchain se puede integrar según el siguiente modelo de casos de uso:

- Usuario Control de Producción.
  Este actor realiza dos casos de uso. En el primero utiliza el interfaz web para registrar en el sistema de la Organización cliente las vicisitudes de la petición de mantenimiento de un componente que debe ser sometido a alguna revisión. Cuando se registra el envío al centro reparador se inicia una transacción en la blockchain indicándolo

  El segundo caso de uso describe al usuario consultando a la blockchain el estado de un mantenimiento que se está realizando en el Centro Reparador.

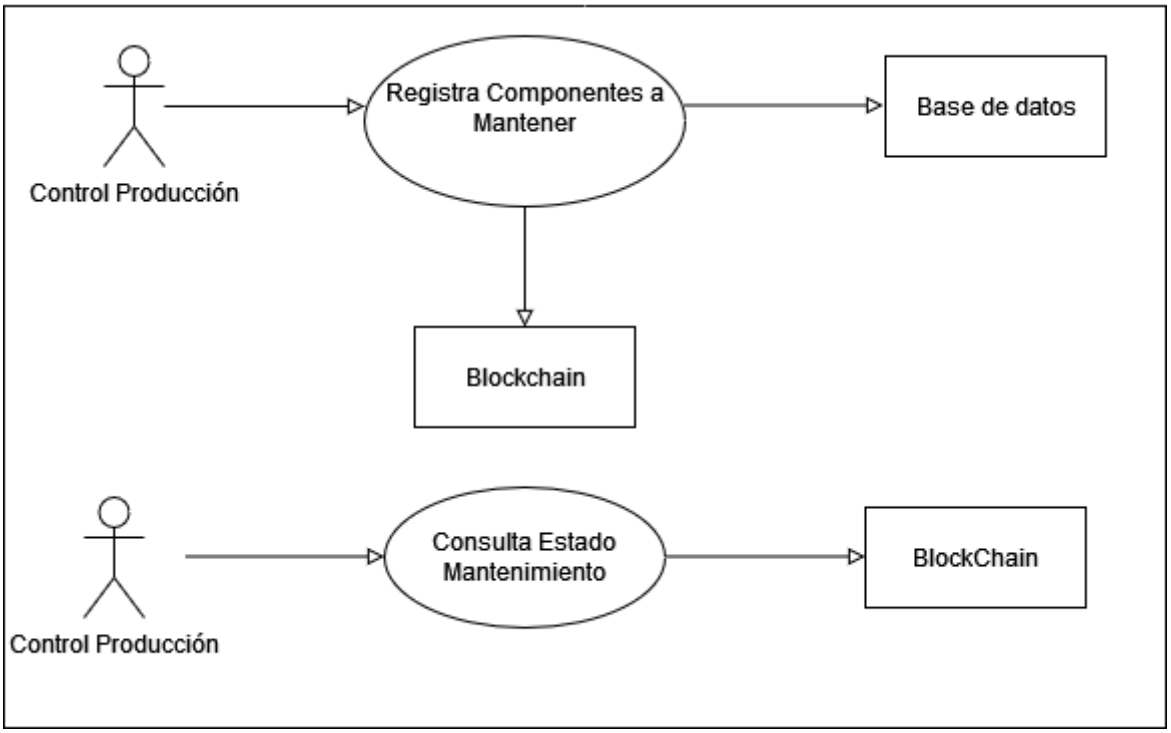

Figura 12: Diagrama de casos de uso organización cliente con blockchain integrada, (fuente: elaboración propia).

- Usuario Operador.
  Este actor realiza dos casos de usos En el primer caso de uso se describe al usuario consultando la información que contiene la blockchain acerca de un equipo en concreto. En el segundo caso de uso el usuario registra en la blockchain las actuaciones realizadas sobre el equipo en la blockchain compartida.

  Este operador es una metáfora de varios roles que existen en el Centro Reparador: el transportista que recoge el equipo en las instalaciones de la organización cliente, el administrativo del almacén que controla los equipos o el mecánico que repara los equipos en el taller. Todos realizan la misma acción de consultar la blockchain y registran la labor que realizan.

1.6.2. Casos de uso preliminares. Organización Centro Reparador.

El centro reparador tiene un actor operador que realiza dos casos de uso:

- En el primero consulta los equipos a mantener en la blockchain así como la tarea a realizar sobre ellos.
- En se segundo caso de uso registra el trabajo realizado en un equipo en la blockchain.

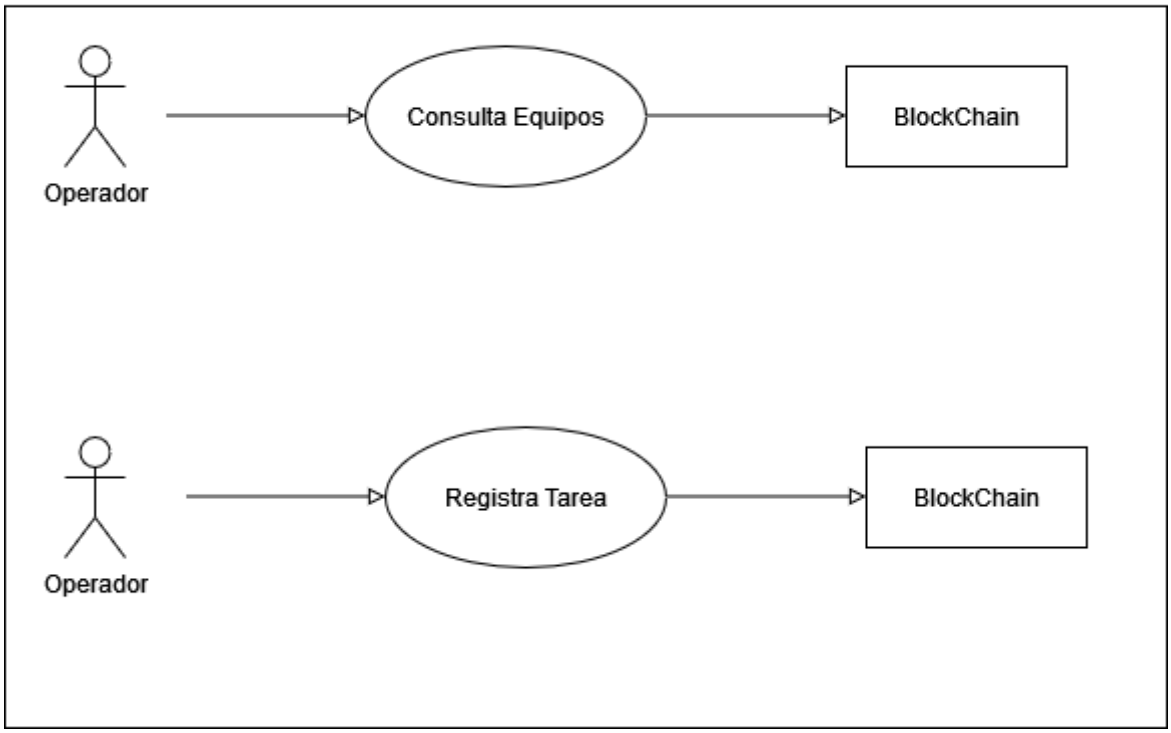

Figura 13: Diagrama de casos de uso organización centro reparador externo con blockchain integrada, (fuente: elaboración propia).

1.6.3. Características y funciones.

Teniendo en cuanta los diagramas de casos de uso se desprenden una serie de requisitos normales (aquellos objetivos fijados con el cliente que producen su satisfacción si están presentes) (Pressman, 2010e) que la aplicación debe tener:

Caso de uso	Organización	Actor	Funcionalidad
CU-1	Cliente	Control de producción	Añadir peticiones
CU-2	Cliente	Control de producción	Buscar peticiones
CU-3	Cliente	Control de producción	Actualizar peticiones
CU-4	Cliente	Control de producción/operario	Enviar equipos
CU-5	Cliente	Control de producción/operario	Recibir equipos
CU-6	Cliente	Control de producción	Registrar equipos
CU-7	Cliente	Control de producción	Consultar estado mantenimiento
CU-8	Centro Reparador	Operador	Consultar equipos
CU-9	Centro Reparador	Operador	Registrar tarea mantenimiento
CU-10	Cliente	Control de producción	Consultar listado peticiones

Figura 14. Trazabilidad caso de uso-actor-funcionalidad, (fuente: elaboración propia).

En la Organización Cliente, el equipo que realiza este trabajo está compuesto por veinte personas acostumbradas al manejo de soluciones informáticas estándar: procesadores de texto, hojas de cálculo, gestión del tiempo, etc. Dentro de la organización son los únicos usuarios que van a utilizar la aplicación,

que no se va a contectar al resto de sistemas de la empresa. Realizan más funciones administrativas además de la gestion del mantenimiento, asi que el nivel de concurrencia en el uso de la aplicación se estima bajo.

En el Centro Reparador el personal administrativo que realiza los registros de información del material en mantenimiento tiene características similares y su uso de la aplicación también es bajo.

1.6.4. Arquitectura preliminar.

Teniendo encuenta la información recopilada la arquitectura elegida es cliente-servidor.

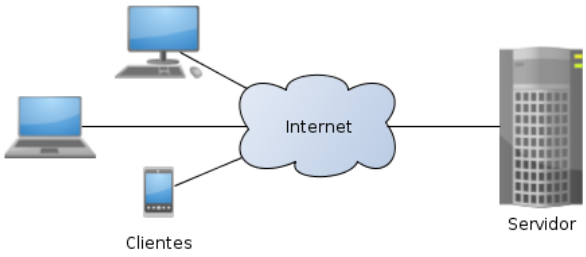

Figura 15: Arquitectura Cliente- Servidor, fuente (De Tiago, 2013).

De esta forma es posible desplegar la aplicación en la red de la empresa o bien en un servidor en internet si es necesario. Cualquier usuario con acceso a un ordenador puede acceder a la interfaz de usuario y consultar, actualizar o añadir información.

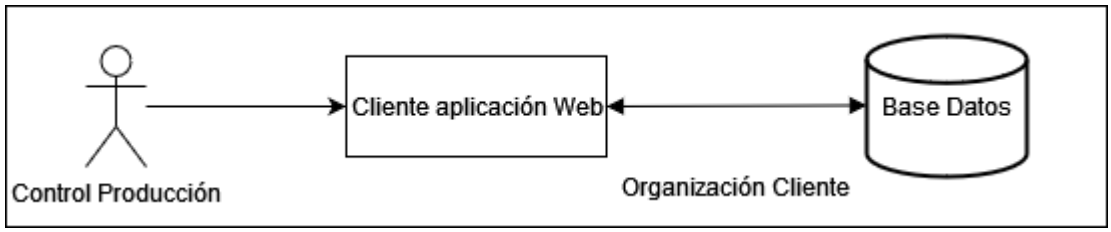

Figura 16: Esquema de uso del sistema, (fuente: elaboración propia).

El usuario Control de Producción registra una petición de mantenimiento a un Centro Reparador Externo utilizando el interfaz web de la aplicación y negocia los términos contractuales de la misma utilizando métodos de comunicación tradicionales (correo electrónico, teléfono, reuniones, etc.)

Cuando el equipo se entrega al Centro Reparador se registra su envío.

Una vez el equipo vuelve a las instalaciones de la Organización Cliente, el usuario Control de Producción cierra la petición de mantenimiento registrándolo a través de la interfaz web para realizar su facturación y pago si está todo correcto.

La organización centro reparador sigue el mismo esquema de uso, con la diferencia que, su sistema recibe la petición de mantenimiento y el operador registra el mantenimiento realizado. La comunicación entre ambas organizaciones se realiza a través del registro de las operaciones en la red blockchain.

1.6.5. Planeamiento.

Para el desarrollo de una primera versión del proyecto se ha realizado un cálculo de las horas necesarias que se puede observar en el siguiente cuadro resumen:

Proyecto	Horas
Documentación Tecnologías	30
Especificación Requisitos	15
Análisis de alto nivel	10
Diseño del sistema	45
Implementación y pruebas	80
Documentación	30
Total Horas del Proyecto	210

Figura 17: Previsión de las horas por tareas, (fuente elaboración propia).

- Documentación tecnologías a aplicar: 30 horas
  Spring Web y MySQL 10 horas.
  Hyperledger FireFly, Hyperledger Fabric y AWS Managed Blockchain: 20 horas.

- Especificación de requisitos. 15 horas.
  Definición de casos de uso. Trazabilidad de las funcionalidades.

- Análisis de alto nivel. 10 horas
  Definición arquitectura del sistema. Descomposición del proyecto. Refinamiento y modelado de los casos de uso.

- Diseño del sistema: 45 horas.
  Front: 15 horas.
  Backend: 20 horas.
  Red Blockchain: 20 horas

- Implementación y pruebas. 80 horas
  Front: 20 horas
  Backend: 40 horas
  Blockchain: 40 horas

- Documentación. 30 horas
  Manual Usuario y administración. 10 horas
  Memoria TFM. 20 horas

El riesgo principal del proyecto surge del desconocimiento de partida de las nuevas tecnologías a aplicar: Spring Boot, MySQL, AWS Managed Blockchain, Hyperledger FireFly e Hyperledger Fabric. El desconocimiento de partida sobre su configuración, gestión, despliegue e interacción con un proyecto

clásico front-backend puede dificultar su integración con él y hacer que el tiempo dedicado finalmente al proyecto sea mayor al previsto.

Se planifican 52 días de trabajo a razón de 4 horas diarias.

Se dispone de un ordenador portátil conectado a internet HP Pavilion 16 (pantalla 16", procesador i7, 16 Gb Ram, 1Tb SSD) y una pantalla externa HP 24fh.

1.7. Fase de elaboración.

En esta fase continua la colaboración con el cliente para mejorar y ampliar los casos de uso preliminares. Con la nueva información se refina la representación de la arquitectura, modelándola con hasta cinco puntos de vista diferentes del software: modelos de casos de uso, de requisitos, modelos de diseño, implementación y despliegue. En algunos proyectos esto genera una "línea base de la arquitectura ejecutable" (Arlow, 2002) que demuestra su viabilidad, aunque no proporcione toda la funcionalidad del sistema.

Dada la mejor comprensión del proyecto a desarrollar, es recomendable en esa fase revisar el plan para asegurar que su alcance y fechas de entrega iniciales, continúan siendo razonables (Pressman, 2010b).

1.7.1. Refinamiento casos de uso.

Un caso de uso describe la forma en la que el usuario del sistema interactúa con el software para conseguir un objetivo. Describe los actores que intervienen, sus objetivos, las precondiciones del sistema, el escenario en el que se desarrolla, su prioridad o las excepciones que pueden ocurrir (Pressman, 2010f). Toda esta información junto a un diagrama de casos de uso facilita la comprensión del comportamiento esperado de la aplicación.

A continuación, se narra cada caso de uso junto a su diagrama.

1.7.1.1. CU-1. Añadir petición de mantenimiento.

Caso de Uso: Añadir Petición de mantenimiento.

Actor principal: Control de producción.

Objetivo: Añadir una nueva petición de mantenimiento con la información de un equipo que es necesario realizarle algún tipo de mantenimiento.

Precondición: La petición no debe existir en el sistema de la organización cliente. El equipo debe existir en el inventario.

Disparador: Control de producción necesita realizar una tarea de mantenimiento a algún tipo de material.

Escenario:

1. Control de producción accede a la pantalla principal.
2. Control de producción selecciona añadir petición de mantenimiento.
3. Control de producción introduce información en pantalla de añadir petición.
4. Control de producción: observa la información de la nueva petición y la graba en el sistema.
5. Control de producción: Añade un equipo a la petición de mantenimiento.

Excepción:

1. Petición de mantenimiento está en el sistema. Control de producción observa la información que le muestra el sistema y la comprueba.

2. Petición de mantenimiento con dato equivocado. Control de producción realiza de nuevo el proceso de añadir con información corregida.
3. Equipo no existe. Control de producción lo añade al inventario.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 1.

Diagrama:

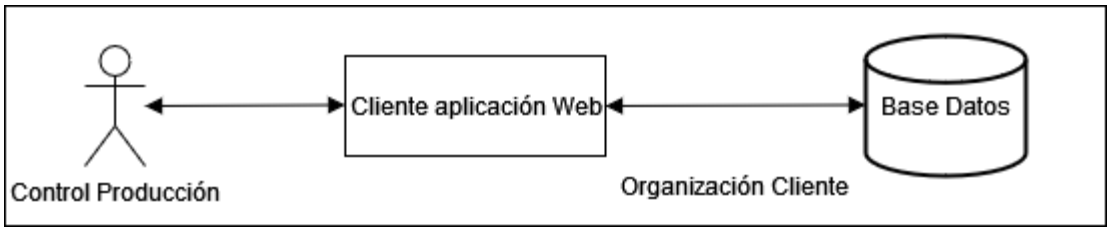

Figura 18: Diagrama caso de uso 1, (fuente: elaboración propia).

1.7.1.2. CU-2. Buscar petición de mantenimiento.

Caso de Uso: Buscar Petición de mantenimiento.

Actor principal: Control de producción.

Objetivo: Buscar una petición de mantenimiento para obtener información de un equipo a mantener o actualizar la información que contiene.

Precondición: La petición debe existir en el sistema de la organización cliente.

Disparador: Control de producción necesita información de una petición o bien actualizarla.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona buscar petición de mantenimiento.
3. Control de producción: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción: observa la información de la petición que el sistema le muestra.

Excepción:

1. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.
2. Petición de mantenimiento no está en el sistema comprobada información de búsqueda. Control de producción introduce nueva petición en el sistema.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 1.

Diagrama:

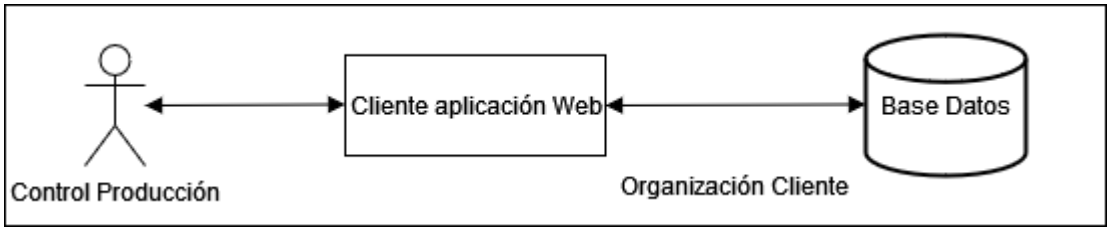

Figura 19: Diagrama caso de uso 2, (fuente: elaboración propia).

1.7.1.3. CU-3. Actualizar petición de mantenimiento.

Caso de Uso: Actualizar Petición de mantenimiento.

Actor principal: Control de producción.

Objetivo: Actualizar o añadir información referida a un equipo a mantener.

Precondición: La petición debe existir en el sistema de la organización cliente.

Disparador: Control de producción necesita actualizar o añadir información de un equipo que necesita mantenimiento.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona buscar petición de mantenimiento.
3. Control de producción: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción: observa la información de la petición que el sistema le muestra y actualiza o añade información.
5. Control de producción: Graba la información en el sistema.

Excepción:

1. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.
2. Petición de mantenimiento no está en el sistema comprobada información de búsqueda. Control de producción no puede enviar el equipo y comunica a centro de control por correo los datos del equipo físico para investigación de las causas.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 1.

Diagrama:

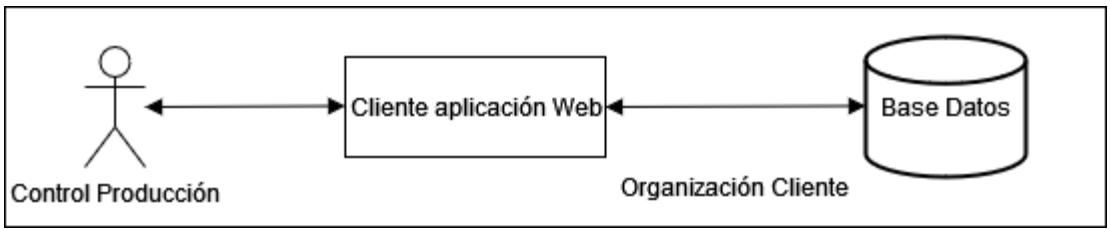

Figura 20: Diagrama de caso de uso 2, (fuente: elaboración propia).

1.7.1.4. CU-4. Enviar equipos al centro reparador a mantener.

Caso de Uso: Enviar Equipos.

Actor principal: Control de producción/operario.

Objetivo: Registrar el envío de un equipo al centro reparador porque ha venido a la organización cliente a recogerlo.

Precondición: La petición debe existir en el sistema de la organización cliente.

Disparador: El operador logístico del Centro Reparador se ha desplazado a la Organización Cliente a recogerlo.

Escenario:

1. Control de producción/operario: accede a la pantalla principal.
2. Control de producción/operario selecciona buscar petición de mantenimiento.
3. Control de producción/operario: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción/operario: observa la información de la petición que el sistema le muestra y añade la fecha de envío.
5. Control de producción/operario: Graba la información en el sistema.

Excepción:

1. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.
2. Petición de mantenimiento no está en el sistema comprobada información de búsqueda. Control de producción/operario no puede enviar el equipo y comunica a centro de control por correo los datos del equipo físico para investigación de las causas.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 1.

Aspectos pendientes:

1. Mostrar formulario para rellenar información del equipo físico y enviarla a centro de control automáticamente para ser subsanado el error.

Diagrama:

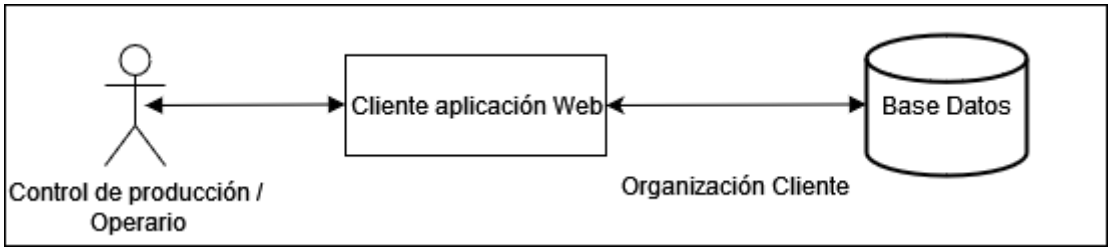

Figura 21: Diagrama de caso de uso 4, (fuente: elaboración propia).

1.7.1.5. CU-5. Recibir equipos en organización cliente.

Caso de Uso: Recibir Equipos.

Actor principal: Control de producción/operario.

Objetivo: Registrar la recepción de un equipo que ha vuelto a la organización cliente desde el centro reparador.

Precondición: La petición debe existir en el sistema de la organización cliente.

Disparador: El operador logístico del Centro Reparador se ha desplazado a la Organización Cliente a devolver el equipo.

Escenario:

1. Control de producción/operario: accede a la pantalla principal.
2. Control de producción/operario selecciona buscar petición de mantenimiento.
3. Control de producción/operario: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción/operario: observa la información de la petición que el sistema le muestra y añade la fecha de recepción.
5. Control de producción/operario: Graba la información en el sistema.

Excepción:

1. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.
2. Petición de mantenimiento no está en el sistema comprobada información de búsqueda. Control de producción/operario no puede recepcionar el equipo, y comunica a centro de control por correo los datos del equipo físico para investigación de las causas. El operador logístico se vuelve a llevar el equipo.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 1.

Aspectos pendientes:

1. Mostrar formulario para rellenar información del equipo físico y enviarla a centro de control y a la organización Centro Reparador automáticamente para ser subsanado el error.

Diagrama:

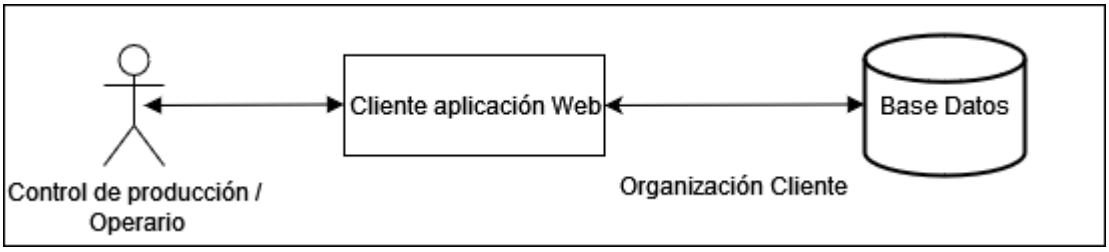

Figura 22: Diagrama de caso de uso 5, (fuente: elaboración propia).

1.7.1.6. CU-6. Registrar equipos.

Caso de Uso: Registrar Equipos.

Actor principal: Control de producción.

Objetivo: Registra el equipo a mantener en la red blockchain para conocimiento del resto de miembros interesados.

Precondición: La petición debe existir en el sistema de la organización cliente.

Disparador: Control de producción se ha puesto de acuerdo con el centro reparador en las condiciones contractuales del mantenimiento a realizar y lo registra en red..

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona buscar petición de mantenimiento.
3. Control de producción: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción: observa la información de la petición que el sistema le muestra y actualiza la información contractual.
5. Control de producción: Graba la información en el sistema y se dispara el proceso de registro de la información en la red blockchain.

Excepción:

1. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 2.

Diagrama:

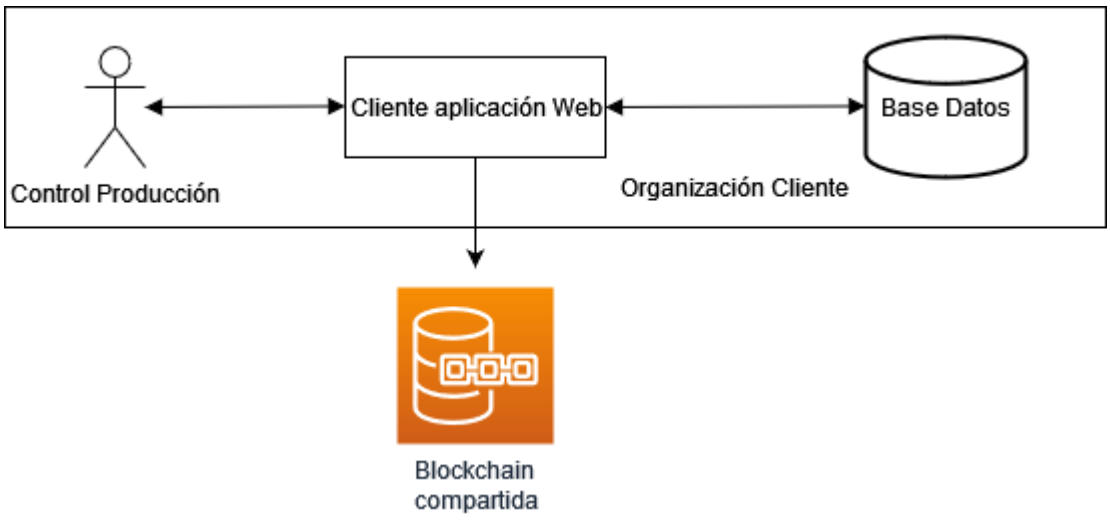

Figura 23: Diagrama de caso de uso 6, (fuente: elaboración propia).

1.7.1.7. CU-7. Consultar estado del mantenimiento de un equipo.

Caso de Uso: Consultar estado de mantenimiento.

Actor principal: Control de producción.

Objetivo: Actualizar la última información de un equipo en mantenimiento en el centro reparador (típicamente son inspección de entrada, trabajando, inspección de salida, operativo o inoperativo).

Precondición: La petición debe existir en el sistema de la organización cliente y haberse enviado al centro reparador.

Disparador: Control de producción necesita información del estado de mantenimiento de un equipo.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona buscar petición de mantenimiento.
3. Control de producción: introduce información para búsqueda en pantalla de búsqueda.
4. Control de producción: observa la información de la petición que el sistema le muestra.
5. Control de producción: Actualiza la información en su sistema desde el centro reparador.

Excepción:

3. Petición de mantenimiento no está en el sistema. Control de producción comprueba información de búsqueda y repite el proceso.
4. Centro reparador no ha rellenado el estado. Ponerse en contacto con centro reparador.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 2.

Aspectos pendientes:

1. Implementar sistema de mensajes con organización centro reparador.

Diagrama:

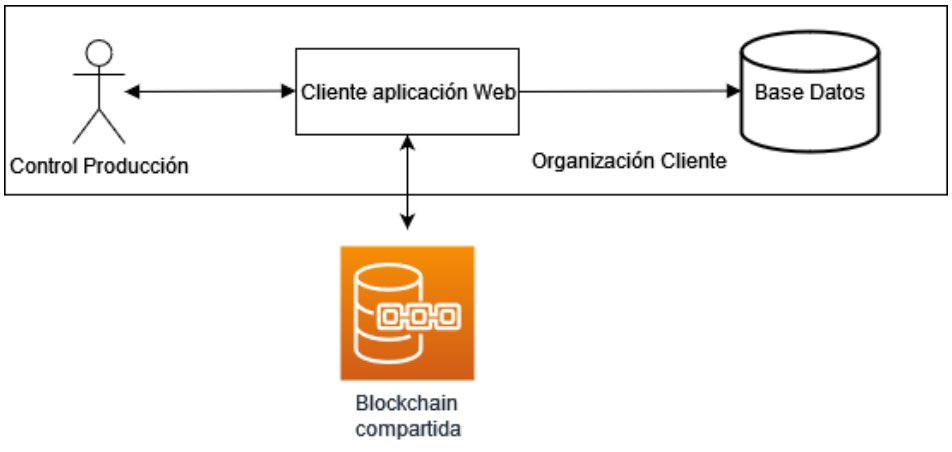

Figura 24: Diagrama de caso de uso 7, (fuente: elaboración propia).

1.7.1.8. CU-8. Consultar un equipo.

Caso de Uso: Consultar equipo.

Actor principal: Operador.

Objetivo: Consultar la información de un equipo que está manteniéndose en el centro reparador

Precondición: La información debe existir en el sistema del centro reparador.

Disparador: el centro reparador necesita información del equipo.

Escenario:

1. Operador: accede a la pantalla principal.
2. Operador: selecciona buscar equipo.
3. Operador: introduce información para búsqueda en pantalla de búsqueda.
4. Operador: observa la información el sistema le muestra.

Excepción:

1. Equipo no está en el sistema. Operador comprueba información de búsqueda y repite el proceso.
2. Organización cliente no ha registrado el equipo en la red. Ponerse en contacto con organización cliente.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 3.

Aspectos pendientes:

1. Implementar sistema de mensajes con organización cliente.

Diagrama:

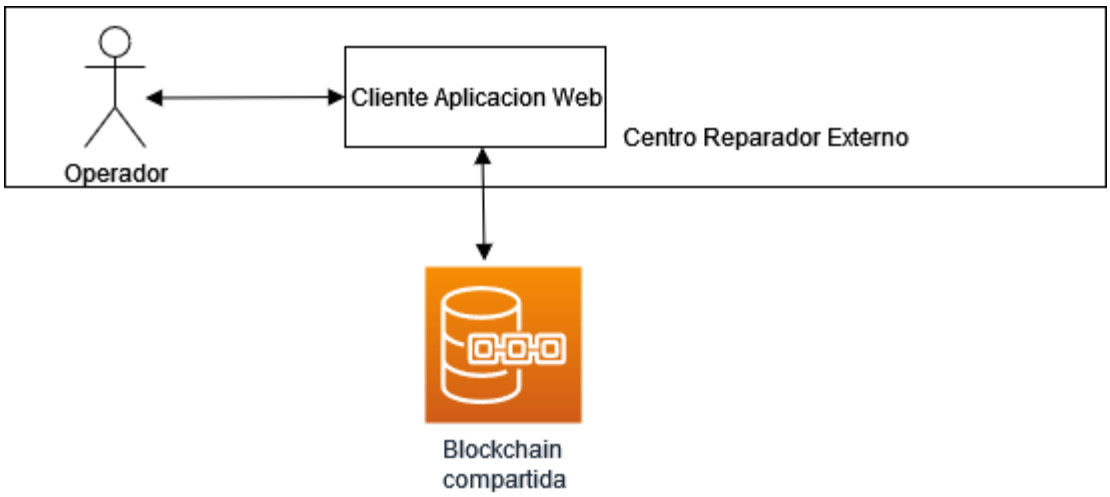

Figura 25: Diagrama de caso de uso 8(fuente: elaboración propia).

1.7.1.9. CU-9. Añadir tarea de mantenimiento a un equipo.

Caso de Uso: Añadir tarea de mantenimiento.

Actor principal: Operador.

Objetivo: Añadir la tarea de mantenimiento que se aplica a un equipo y/o actualizar el estado del manteniento.

Precondición: La información debe existir en el sistema del centro reparador.

Disparador: El centro reparador quiere registrar la tarea de mantenimiento a un equipo o actuarlizar el estado de mantenimiento del equipo.

Escenario:

1. Operador: accede a la pantalla principal.
2. Operador: selecciona buscar equipo.
3. Operador: introduce información para búsqueda en pantalla de búsqueda.
4. Operador: observa la información el sistema le muestra.
5. Operador: Añade la tarea de mantenimiento que va a aplicar o actualiza el estado del mantenimiento del equipo. Típicamente inspección de entrada, trabajando, inspección de salida, operativo o inoperativo.

Excepción:

1. Equipo no está en el sistema. Operador comprueba información de búsqueda y repite el proceso.
2. Organización cliente no ha registrado el equipo en la red. Ponerse en contacto con organización cliente.

Prioridad: Esencial (requisito normal). Debe implementarse.

Disponible:En el incremento 3.

Aspectos pendientes:

1. Implementar sistema de mensajes con organización cliente.

Diagrama:

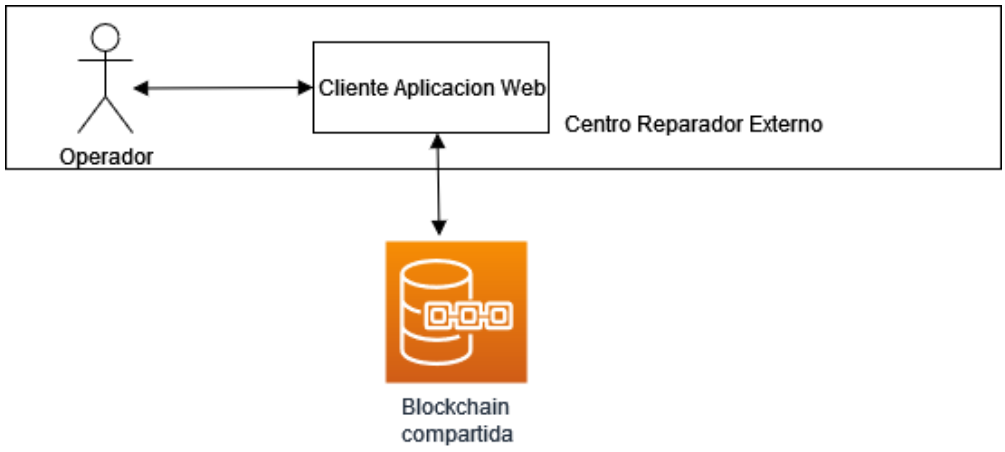

Figura 26: Diagrama de caso de uso 9, (fuente: elaboración propia).

1.7.1.10. CU-10. Listado de peticiones de mantenimiento.

Caso de Uso: Listar todas las peticiones de mantenimiento.

Actor principal: Control de producción.

Objetivo: El sistema muestre un listado con todas las peticiones del sistema.

Precondición: No hay.

Disparador: Control de producción necesita buscar alguna petición de forma manual.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona listado petición de mantenimiento.
3. Control de producción: observa la información de la petición que el sistema le muestra.

Excepción:

1. No hay.

Prioridad: Esencial (requisito esperado-implícito en el producto-). Debe implementarse.

Disponible: En el incremento 3.

Diagrama:

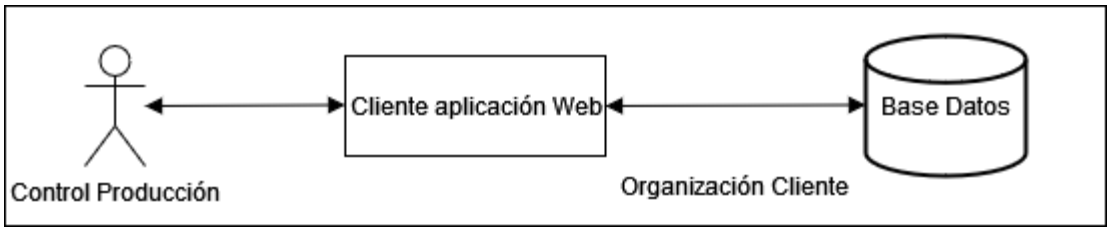

Figura 27: Diagrama de caso de uso 10, fuente elaboración propia.

1.7.1.11. CU-11. Listado de equipos en el inventario.

A partir de proceso de refinamiento de los casos de uso se descubre la necesidad de disponer de control del inventario de los equipos en la organización cliente. Para cumplir las precondiciones de alguno de los casos de uso o bien poder realizar alguna de las excepciones (como en el CU-1) es necesario realizar gestión sobre los equipos. Se añaden tres casos de uso más para este fin.

Caso de Uso: Listar todos los equipos del inventario.

Actor principal: Control de producción.

Objetivo: El sistema muestre un listado con todos los equipos.

Precondición: No hay.

Disparador: Control de producción necesita buscar algún equipo de forma manual.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona el control de inventario.
3. Control de producción: observa la información en el listado que el sistema le muestra.

Excepción:

1. No hay.

Prioridad: Esencial (requisito esperado-implícito en el producto-). Debe implementarse.

Disponible: En el incremento 3.

Diagrama:

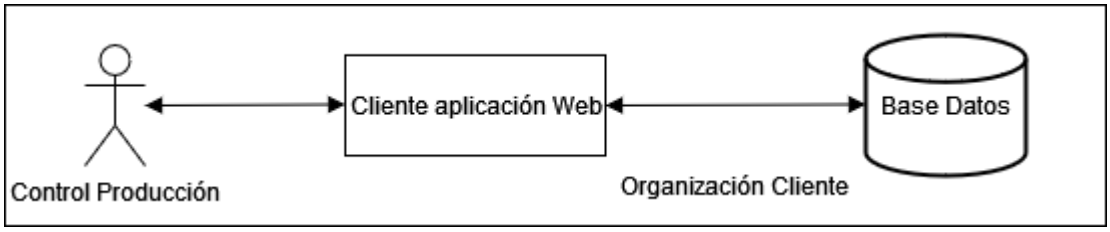

Figura 28: Diagrama de caso de uso 11, fuente elaboración propia.

1.7.1.12. CU-11. Editar/añadir equipos en el inventario.

Caso de Uso: Editar o añadir un equipo del inventario.

Actor principal: Control de producción.

Objetivo: El sistema permita modificar o añadir al inventario información de un equipo.

Precondición: Para poder modificar el equipo debe existir. Para añadir el equipo no debe existir.

Disparador: Control de producción necesita añadir o modificar información de un equipo.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona el control de inventario.
3. Control de producción: observa la información en el listado que el sistema le muestra.
4. Control de producción: Pulsa en el botón editar o añadir nuevo equipo.
5. Control de producción: Completa la información en el formulario que se muestra.
6. Control de producción: Pulsa el botón guardar.

Excepción:

1. No hay.

Prioridad: Esencial (requisito esperado-implícito en el producto-). Debe implementarse.

Disponible: En el incremento 3.

Diagrama:

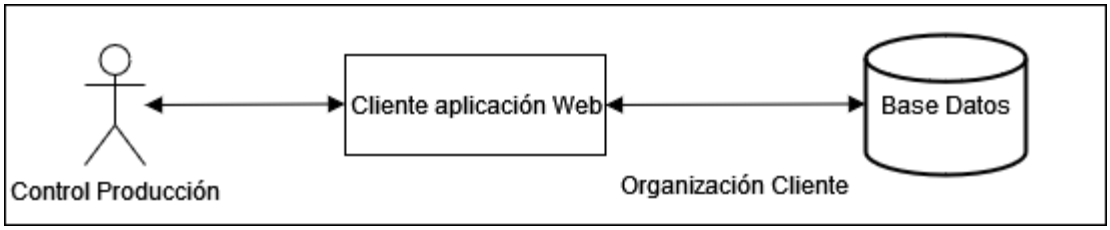

Figura 29: Diagrama de caso de uso 12, fuente elaboración propia.

1.7.1.13. CU-13. Seleccionar equipo para una petición de mantenimiento.

Caso de Uso: Selecciona un equipo del inventario para una petición de mantenimiento.

Actor principal: Control de producción.

Objetivo: El sistema permita elegir un equipo y anexarlo a una petición de mantenimiento.

Precondición: Para añadir el equipo debe existir en el inventario

Disparador: Control de producción necesita añadir un equipo en una petición de mantenimiento.

Escenario:

1. Control de producción: accede a la pantalla principal.
2. Control de producción: selecciona el control de inventario.
3. Control de producción: observa la información en el listado que el sistema le muestra.
4. Control de producción: Pulsa en el botón seleccionar.
5. Control de producción: El sistema pide confirmación que debe ser confirmada.

Excepción:

1. No hay.

Prioridad: Esencial (requisito esperado-implícito en el producto-). Debe implementarse.

Disponible: En el incremento 3.

Diagrama:

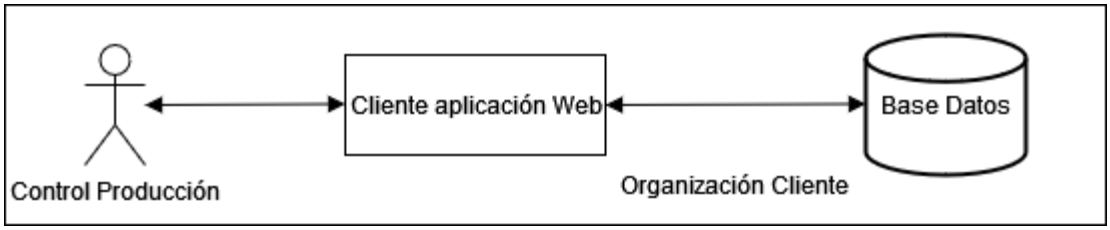

Figura 30: Diagrama de caso de uso 13, fuente elaboración propia.

Caso de uso	Organización	Actor	Funcionalidad
CU-1	Cliente	Control de producción	Añadir peticiones
CU-2	Cliente	Control de producción	Buscar peticiones
CU-3	Cliente	Control de producción	Actualizar peticiones
CU-4	Cliente	Control de producción/operario	Enviar equipos
CU-5	Cliente	Control de producción/operario	Recibir equipos
CU-6	Cliente	Control de producción	Registrar equipos
CU-7	Cliente	Control de producción	Consultar estado mantenimiento
CU-8	Centro Reparador	Operador	Consultar equipos
CU-9	Centro Reparador	Operador	Registrar tarea mantenimiento
CU-10	Cliente	Control de producción	Consultar listado peticiones
CU-11	Cliente	Control de producción	Consultar listado equipos
CU-12	Cliente	Control de producción	añadir/modificar equipos
CU-13	Cliente	Control de producción	seleccionar petición/equipo

Figura 31: Tabla Casos de uso refinados-funcionalidad, fuente elaboración propia.

1.7.2. Modelo de navegación.

Teniendo en cuenta la información que se desprende de los casos de uso, se ha modelado cómo navega cada categoría de usuario de un elemento del cliente web a otro.

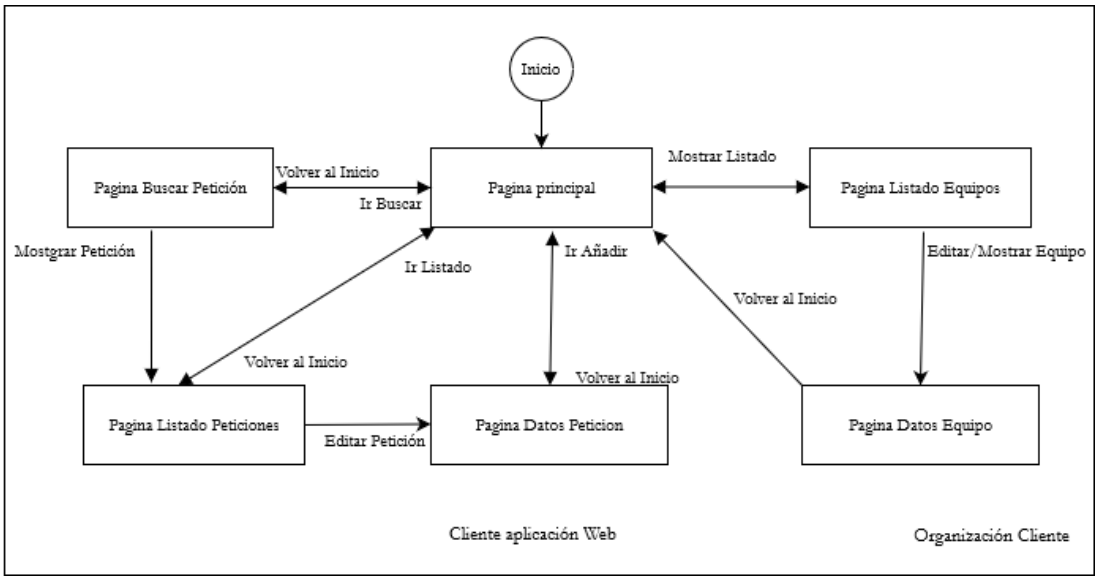

Figura 32: Mapa de navegación web Organización cliente, (fuente: elaboración propia).

Como se puede observar los clientes web finalmente serán muy parecidos, la diferencia está en que el cliente web del Centro Reparador va a obtener la información de la red blockchain partiendo de la identificación del equipo y, a partir de ahí, puede registrar el trabajo de mantenimiento sobre el equipo y registrarlo de nuevo en la blockchain.

La organización cliente, sin embargo, trabaja con las peticiones de mantenimiento y su base de datos donde están almacenadas. Sólo registra la información en la blockchain si se va a realizar el envío a un centro reparador.

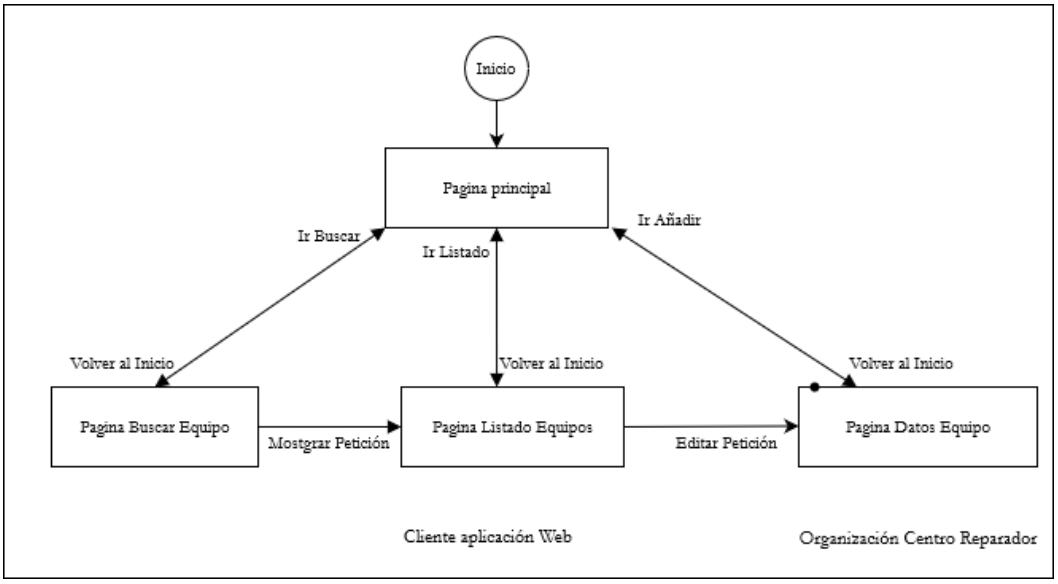

Figura 33: Mapa de navegación web Centro reparador, (fuente: elaboración propia).

Como se puede observar en los casos de uso CU-1, CU-3, CU-4 y otros, son los equipos a mantener la entidad sobre la que se realizan muchas de las operaciones que el sistema debe realizar, por lo que se decide incluir un sistema de control de inventario. Las funciones que este sistema implementará según el refinamiento de los casos de uso debe ser añadir, consultar y actualizar información de los equipos.

1.7.3. Modelo de contenido.

El modelo de contenido incluye los elementos estructurales que agrupan los elementos de contenido, y las entidades que son visibles para el usuario. Estas entidades son las que el usuario va a crear o manipular cuando utilice la aplicación (Pressman, 2010g).

En este caso, los elementos de contenido se agrupan en páginas web y dentro de estas en distintos elementos que agrupan la información. En este caso los objetos de contenido están compuestos de los botones y datos de los equipos a mantener. Para agruparlos se han utilizando los elementos estructurales que el lenguaje de marcado de las páginas web ofrece. En el siguiente punto se trata el modelo de datos que utilizan actualmente y que ayuda a estructurar el contenido de las páginas web.

En este proyecto se han agrupado los objetos de contenido en cuatro páginas web en coherencia con el modelo de navegación: principal o de inicio, búsqueda, listado y por último la página que muestra los datos. Los diseños se han realizado utilizando la herramienta Pencil.

1.7.3.1. Diseño de interfaz de usuario.

La interfaz de usuario comparte un menú superior con enlaces a las páginas que gestionan las peticiones de mantenimiento e indicación de la página donde está situado el usuario.

Los botones utilizados contienen un enlace hacia la página a la que se dirigen o ejecutan una acción en el servidor, por ejemplo, almacenar o borrar. Todos están situados en la parte inferior a la derecha.

Se han utilizado paneles para agrupar la información en categorías y dentro de cada una de izquierda a derecha en función de más a menos importante.

- Pagina de inicio o principal.
  Esta página permite al usuario acceder al listado de peticiones, crear una nueva petición de mantenimiento o bien buscarlas. Dispones para ello de enlaces tanto en el menú superior como botones en los paneles inferiores.

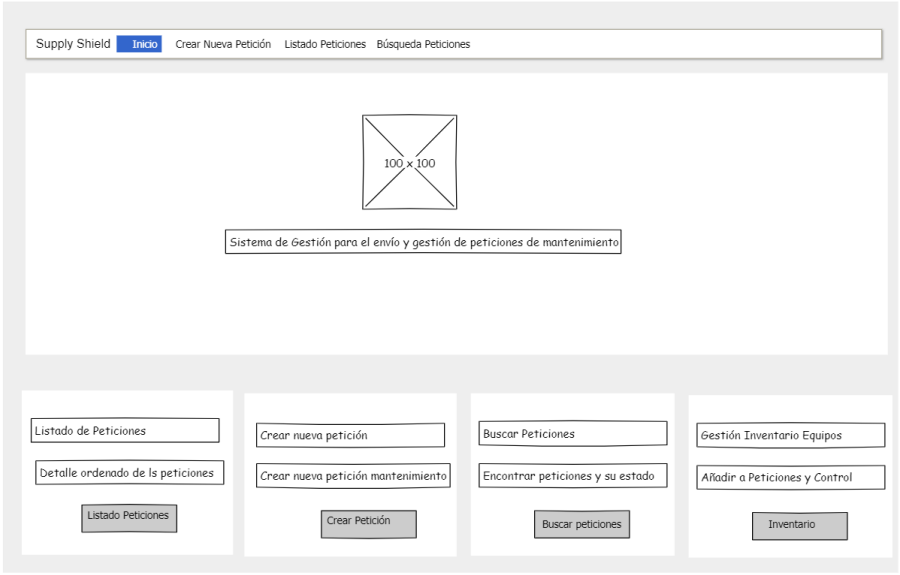

Figura 34: Diseño página principal, (fuente: elaboración propia).

- Página Crear nueva petición.
  Esta página contiene toda la información de una petición de mantenimiento a los largo de su ciclo de vida. Este ciclo tiene su reflejo de arriba hacia abajo y de izquierda a derecha.
  Los botones inferiores permiten bien lanzar un proceso del backend para almacenar información o volver a la página de inicio.

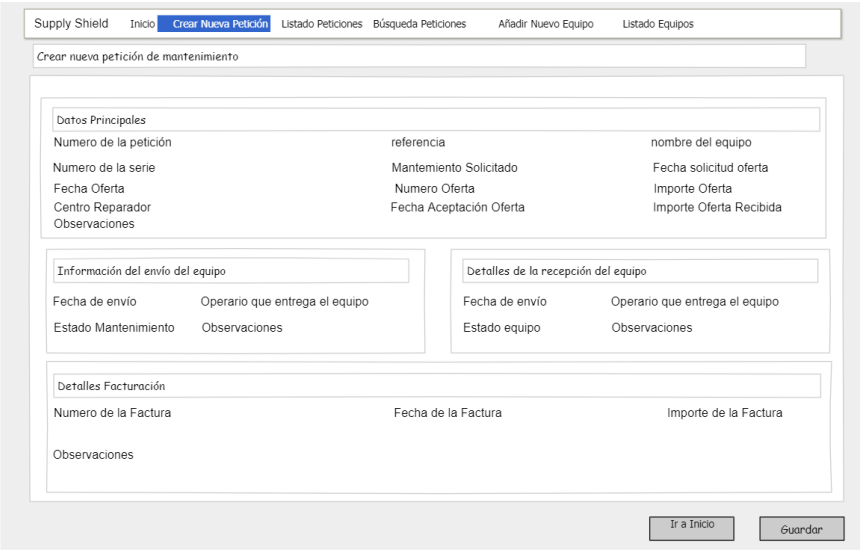

Figura 35: Diseño página Crear Nueva Peticion, (fuente: elaboración propia).

- Página Listado de peticiones.
  Desde esta página es posible bien borrar una petición o editarla para realizar modificaciones a través de dos botones. En el primer caso lanzará un proceso en el backend que borrará la información y en el segundo envía la información a la página "crear peticiones" para mostrarla.

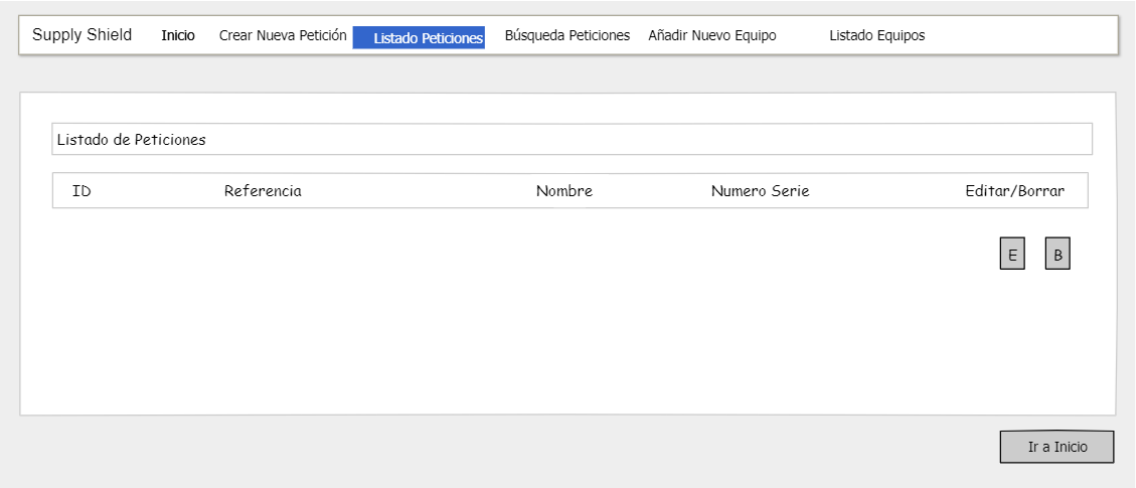

Figura 36: Diseño página Listado de Peticiones, (fuente: elaboración propia).

- Página Buscar peticiones.
  El usuario puede buscar peticiones por la referencia de un equipo o el número de serie. También es posible volver a la página principal a través del menú superior o el botón inferior.

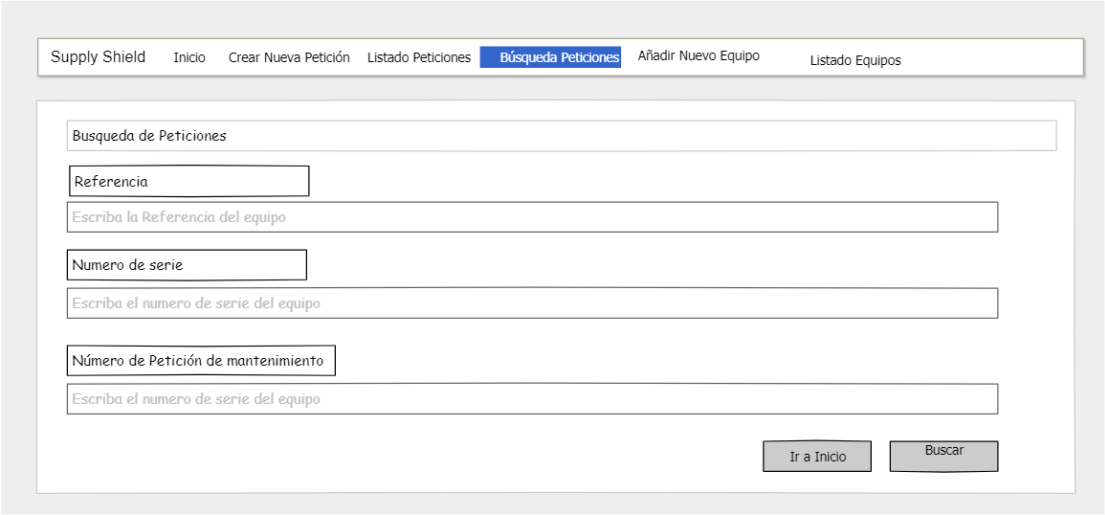

Figura 37: Diseño página Buscar Peticiones, (fuente: elaboración propia).

- Página Listado de equipos.
  El usuario puede desde esta página editar, borrar equipo o añadirlo si es nuevo. Si existe una petición sin equipo también puede asignarle un equipo. También es posible volver a la página principal a través del menú superior o el botón inferior.

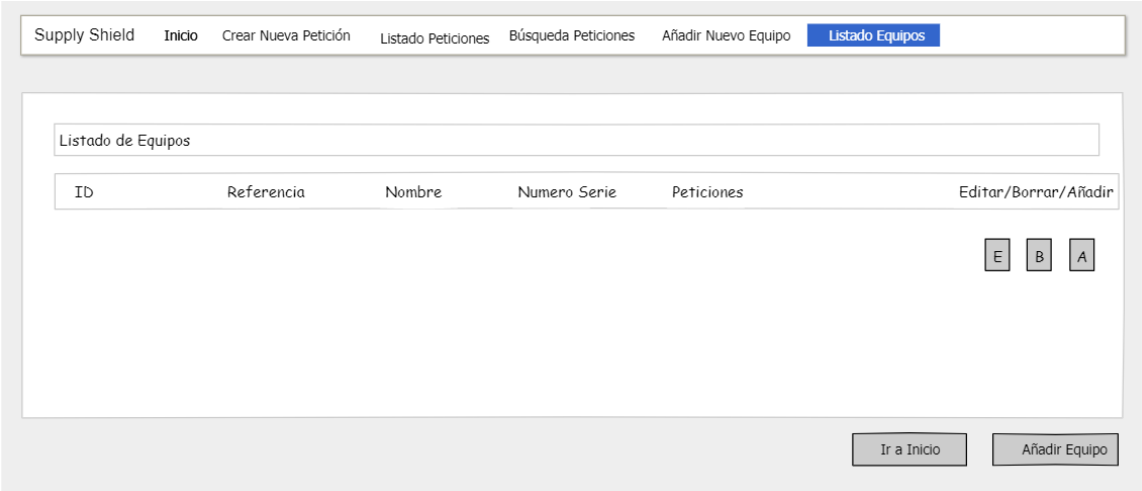

Figura 38: Diseño página Listado de Equipos, (fuente: elaboración propia).

- Página crear nuevo equipo/editar equipo/mostrar información equipo.
  Esta página contiene toda la información de un equipo. Los botones inferiores permiten bien lanzar un proceso del backend para almacenar información, o volver a la página de inicio.

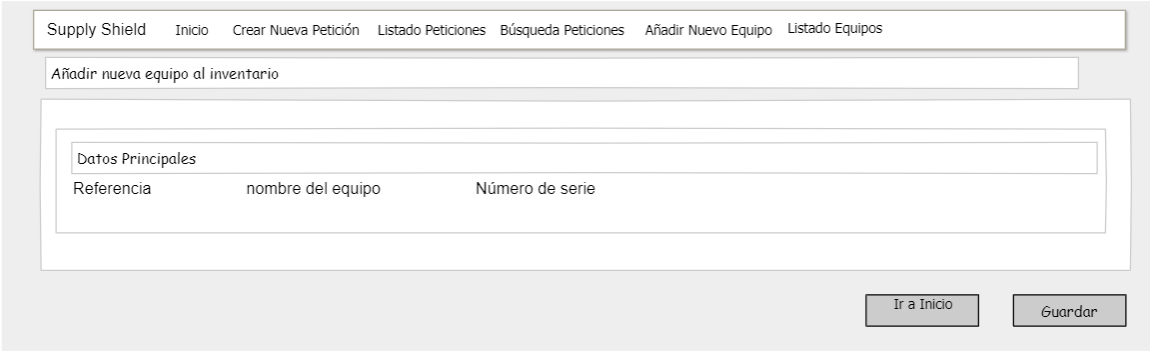

Figura 39: Diseño página Crear nuevo Equipo, (fuente: elaboración propia).

1.7.4. Modelo de datos.

Los usuarios de la aplicación utilizan en la actualidad una hoja Excel en la que almacenan la información de cada petición de mantenimiento en una línea. Dado que los usuarios expresan su interés en que la nueva aplicación mantenga la flexibilidad en los tipos de datos de Excel y la información que ya utilizan, únicamente se añade la nueva información sobre el estado de mantenimiento de un equipo en el centro reparador.

La información que se almacena en ellas está definida con los siguientes campos:

Identificación del equipo y solicitud de oferta.

- Codigo de la petición de mantenimiento.
- Referencia. Es la referencia que le da a un equipo su fabricante.
- Nombre del equipo a mantener.
- Tarea o Mantenimiento solicitado por la organización cliente.
- Fecha de solicitud de oferta. Cuando se solicita a una empresa un servicio y su precio.
- Centro Reparador. Empresa a la que se solicita un Servicio.
- Fecha de la aceptación de la oferta.
- Observaciones de la petición. Campo para ampliar la información si es necesario.

Envío del equipo a la empresa:

- Fecha envío.
- Operario que lo entrega.
- Observaciones del envío. Si necesita contenedores o no. Si ha entregado docuementación etc.
- Estado de mantenimiento. Nuevo campo. Se consulta a la red blockchain para obtenerlo. Típicamente sus valores puede ser inspección de entrada, trabajando, inspección de salida, útil o inútil.
- Hash. Este campo almacena el ultimo hash de la transacción a la blockchain para identificar el equipo.

Recepción del equipo:

- Fecha de la recepción.
- Operario que recibe el equipo.
- Estado del equipo. Util, inútil, pendiente de otra operación de mantenimiento.
- Observaciones de la recepción.

Facturación:

- Numero de factura.
- Fecha de la factura.
- Importe de la factura.
- Observaciones.

Cada uno de los grupos de información se refiere a una de las cuatro fases secuenciales por las que pasa el ciclo de mantenimiento de un componente: se identifica el componente y el mantenimiento necesario para solicitar una oferta al centro reparador, una vez acordados sus términos se envía el equipo, se almacena cuando regresa y si está todo correcto se factura el servicio para cerrar el ciclo. Durante su trabajo los usuarios realizan sobre todo búsquedas por referencia y numero de serie por tener el equipo físicamente delante y ser importante no cometer errores de identificación.

En este trabajo se otorga una gran flexibilidad a la hora de introducir información en los campos, una característica muy apreciada por los usuarios, acostumbrados al uso de hojas excel.

1.7.4.1. Tablas.Modelo E-R.

Para modelizar el almacenamiento de este conjunto de datos se han utilizado dos tablas Equipos y Peticiones. La primera almacenará la información referida a las peticiones de mantenimiento y la segunda la referida a los equipos a los que se realizará alguna tarea de mantenimiento.

A continuación, se puede observar los campos definidos para cada tabla y su tipo. Se han declarado todos campos como cadenas de caracteres, salvo las fechas y los importes con el objetivo de maximizar la flexibilidad del usuario a la hora de introducir información.

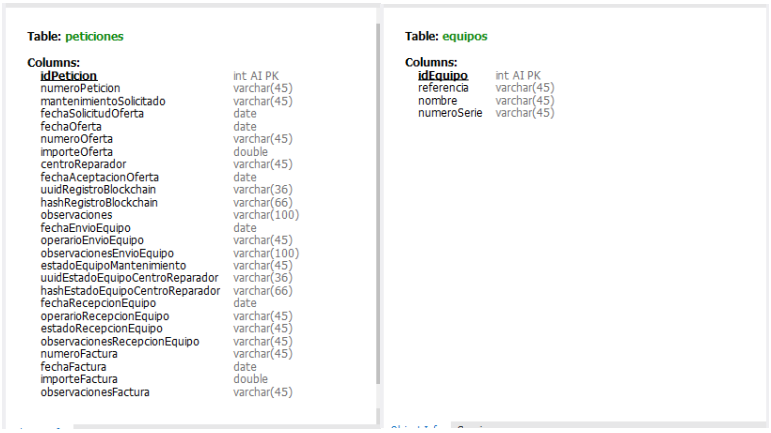

Figura 40: Campos de las tablas peticiones y equipos, (fuente: elaboración propia).

En el inventario de la organización puede haber muchos equipos del mismo tipo que, en principio, van a tener una petición cada uno cuando se envíen a realizarse en ellos algún tipo de mantenimiento. Sin embargo se establece una relación muchos a muchos a través de una nueva tabla peticiones_equipos. Esta tabla solo contiene los campos id_Peticion relacionados con idEquipo. Campos que además son su clave única.

Así es posible conocer no solo las peticiones que se han realizado de un equipo, sino permitir que en una petición pueda enviarse más de un equipo al mismo centro reparador. Entra dentro de lo posible que en el futuro la organización evolucione en ese sentido.

El modelo entidad-relacion puede observarse en el siguientes diagrama:

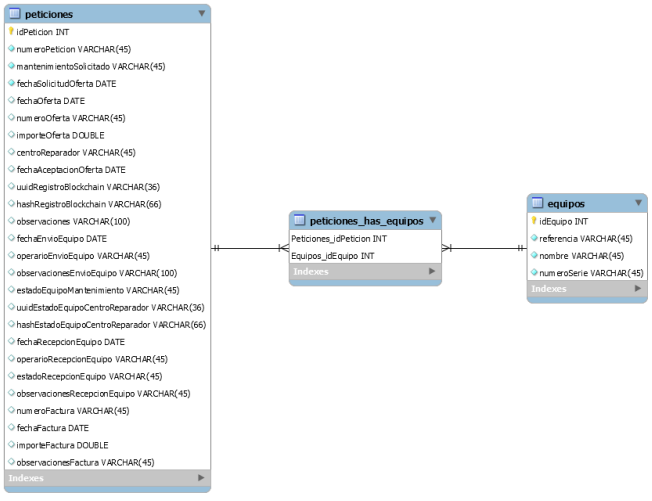

Figura 41: Diagrama Entidad-Relación de la base de datos, (fuente: elaboración propia).

1.7.5. Arquitectura. Modelo de componentes.

La arquitectura de la aplicación es un modelo en tres capas modelo-vista-controlador, que desacopla la interfaz de usuario del comportamiento de la aplicación. El objetivo es facilitar la implementación y la reutilización. El controlador coordina el acceso al modelo, la vista y los datos cruzados entre ellos. El modelo dispone del contenido y realiza el procesamiento. En otras palabras, "La vista es actualizada por el controlador con datos del modelo, basándose en las entradas del usuario" (Pressman, 2010h).

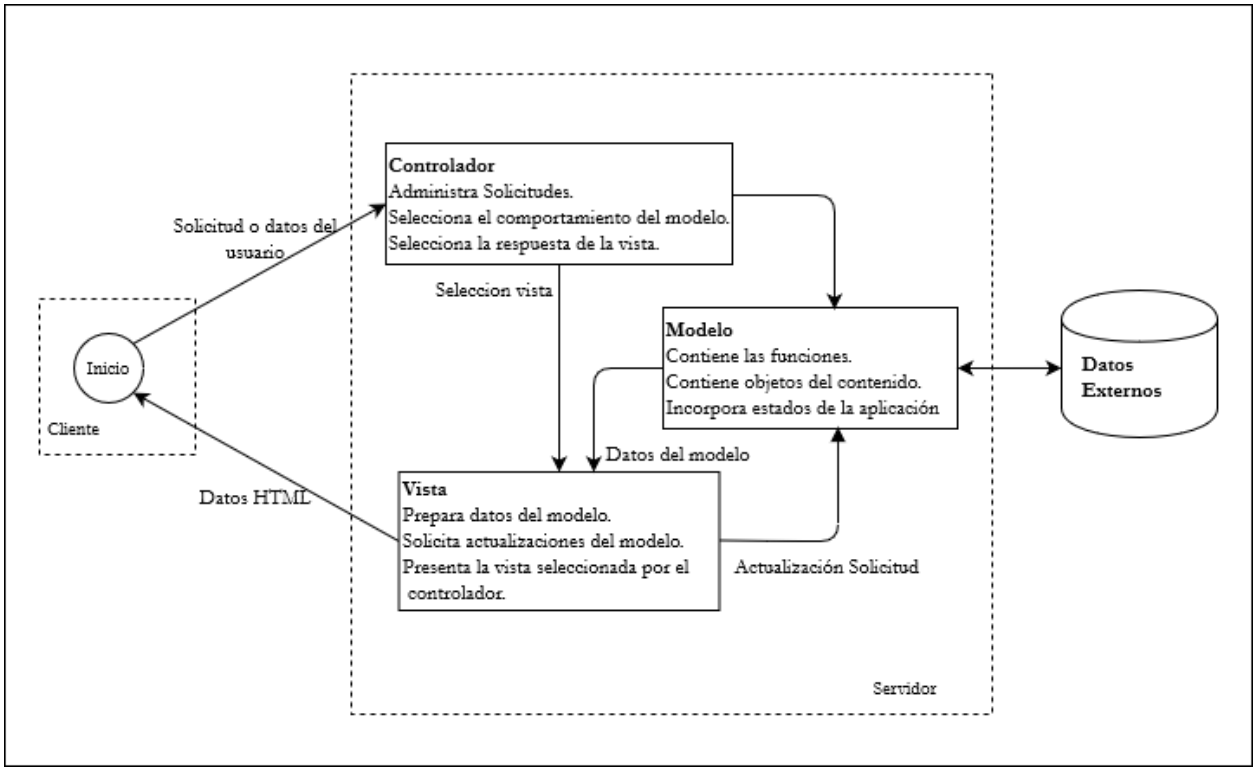

Figura 42: Arquitectura MVC (modelo-vista-controlador) (Pressman, 2010h)

El diseño del cliente web se ha desarrollado en los puntos anteriores. La base de datos o la red blockchain, que actúan como datos externos en el modelo MVC, pueden estar situados en mismo servidor u otro distinto.

Con el ánimo de simplicar la memoria de este trabajo, los componentes utilizados se encuentran descritas en la fase de implementación del proyecto. Son:

- Spring Boot para el desarrollo del cliente web y el modelo MVC del servidor.
- Para el almacenamiento de información una base de datos MySQL.
- Hyperledger FireFly para simular e interactuar con una red blockchain.

1.7.6. Refinamiento del plan del proyecto.

Teniendo en cuenta la ampliación de información de los casos de uso, los modelos de datos de datos y navegación y la arquitectura a implementar, se plantea el siguiente plan del proyecto:

Proyecto	Horas
Documentación Tecnologías	31
Especificación Requisitos	16
Análisis de alto nivel	15
Diseño del sistema	50
Implementación y pruebas	80
Documentación	30
Total Horas del Proyecto	221

Figura 43: Horas planeadas por tareas, (fuente: elaboración propia).

- Documentación tecnologías a aplicar: 30 horas
  Spring Boot y MySQL 10 horas.
  Hyperledger FireFly, Hyperledger Fabric y AWS Managed Blockchain: 20 horas.

- Especificación de requisitos. 15 horas.
  Definición de casos de uso. Trazabilidad de las funcionalidades.

- Análisis de alto nivel. 15 horas
  Definición arquitectura del sistema. Descomposición del proyecto. Refinamiento y modelado de los casos de uso.

- Diseño del sistema: 50 horas.
  Front: 15 horas.
  Backend: 15 horas.
  Red Blockchain: 20 horas

- Implementación y pruebas. 80 horas
  Front: 20 horas
  Backend: 30 horas
  Blockchain: 30 horas

- Documentación. 30 horas
  Manual Usuario y administración. 10 horas
  Memoria TFM. 20 horas

El diagrama de Gannt proyectado teniendo en cuenta las tareas a realizar:

Tareas	May				June				July				Aug				Sept		Horas Tareas
	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	W3	W4	W1	W2	
			Inicio		Elaboración				Construcción							Despliegue			
Documentación			16			10			4							1			31
Requisitos				10			6												16
Modelado				6			6						3						15
Diseño					7			23	12				8						50
Implementacion					8			3		25				25			2		63
Pruebas											8				8		1		17
Documentación												16				10	4		30
			Iter 1			Iter 2			Iter 3				Iter 4			Iter 5			

Figura 44: Diagrama de Gannt, (fuente: elaboración propia).

1.8. Fase de Construcción.

Teniendo en cuenta los casos de uso descritos, las entidades que intervienen en ellos y la arquitectura del proyecto se ha decidido dividirlo en tres partes.

- Una aplicación situada en un servidor backend con una base datos que almacena la información de los equipos a mantener, su lógica de negocio y la base de datos.

- Un cliente web que se encarga de interactuar con el usuario a través de una interfaz que le permita solicitar al backend la siguiente funcionalidad:
  - Registrar los datos de una petición de mantenimiento.
  - Consultar una petición de mantenimiento.
  - Registrar el envío o recepción de un equipo a mantener.
  - Registrar en la blockchain compartida, un equipo que se va a enviar a mantener.
  - Consultar el estado de un equipo en mantenimiento.
  - Añadir, consultar o actualizar información de un equipo en inventario.
- Una red blockchain que se encarga de almacenar la información que se registra en ella de un equipo:
  - Registrar los datos de un equipo y el mantenimiento solicitado/realizado.
  - Consultar un equipo y su estado de mantenimiento.
  - Registrar el estado final del mantenimiento y la tarea realizada.
  - Añadir, consultar, actualizar un equipo o su información asociada.

1.8.1. Desarrollo del backend.

El servidor interactúa de dos formas. Por un lado, ejecuta las consultas procedentes del cliente web contra la base de datos, procesa la lógica de negocio necesaria y entrega la información al cliente web para que la muestre al usuario. Dispone para ello de una serie de peticiones get, post o put que se ejecutan en función de los solicitado por el cliente web.

Por otro el servidor almacena la información de una transacción en la red blockchain cuando las condiciones de la lógica de negocio sean las adecuadas. Por ejemplo, cuando un equipo se envía o recibe a otra empresa, o cuando recibe un evento de la red comuncándole una actualización del estado de mantenimiento de un equipo.

Para implementar estas funcionalidades se ha elaborado un modelo de datos implementado en una base de datos MySQL, a la que se accede a través de objetos JPA proporcionados por las clases que Spring Boot ofrece.

Por otro lado se ha implementado un modelo con las capas que permiten a la base de datos y exponer una API para comunicarse con el cliente web y la red blockchain. Implementación del backend.

Para la implementación del backend se ha desarrollado un microservicio que utiliza varios módulos del framework Spring:

- Spring Boot DevTools para que reinicie de forma automática la aplicación cuando se realizan modificaciones en el código.
- Spring Web para implementar el patrón MVC y que la aplicación exponga un API RESTful.
- Spring Data JPA para dar persistencia los datos y facilitar su acceso a través de JPA.
- MySQL Driver para disponer de un driver jdbc necesario para poder gestionar la base de datos MySQL.

Para ello se ha utilizado Spring Initializr al que se accede desde la web https://start.spring.io, cuya configuración se puede observer en la siguientes imagen:

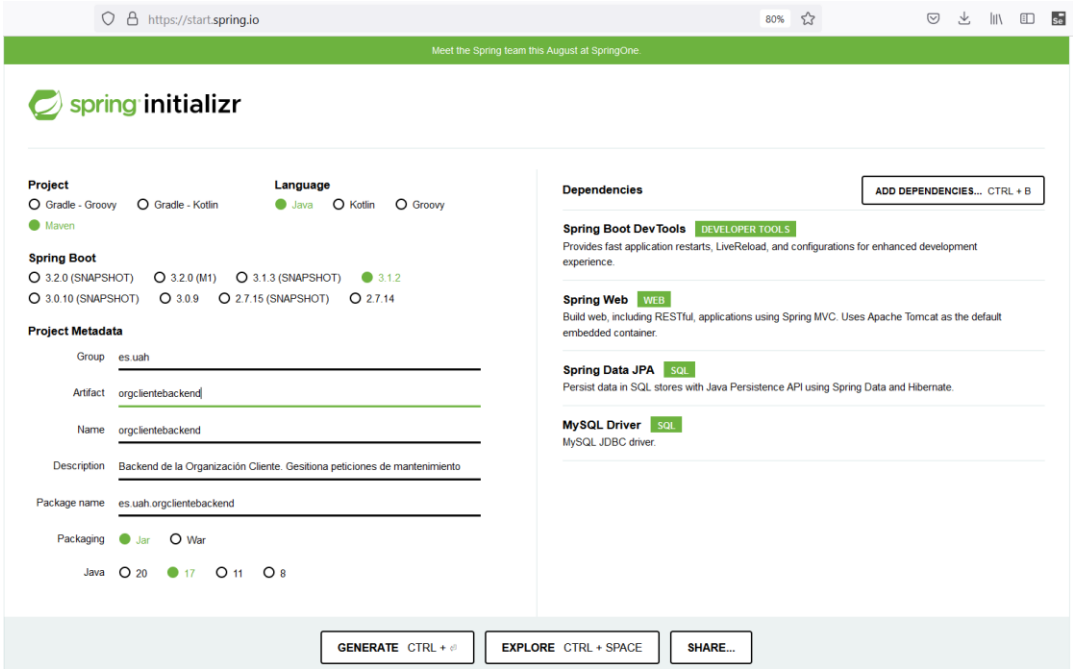

Figura 45: Selección de componentes microservicio backend, (fuente: elaboración propia).

Al pulsar el botón Generate, se descarga el código que servirá de base para el desarrollo de la aplicación. Incluye las dependencias necesarias y las configuraciones básicas para comenzar el desarrollo del microservicio.

Se ha utilizado el entorno de programación IntelliJ IDEA 2023.2.1 Ultimate Edition y jdk 17.

- Descripción de las clases.

Se ha modelizado en tres capas que contienen las clases que componen el patrón DAO para el acceso a datos, el servicio que contiene la lógica de la aplicación y el controlador con las rutas que expone una API.

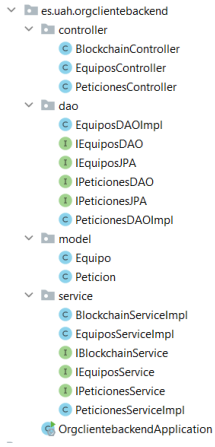

Figura 46: Cojunto de clases que forman la capas del backend, (fuente: elaboración propia).

- DAO.
  En este paquete se sitúan las clases necesarias el acceso a la base de datos. Se utiliza el patron DAO (Data Access Object) para encapsular el acceso a los datos y separarlo de la lógica de negocio.
    - Las clases IPeticionesJPA e IEquiposJPA son interfaces que heredan de JpaRepository las operaciones estándar que se van a realizar sobre los datos de tipo Peticion y Equipo respectivamente. También declaran las operaciones específicas a realizar no soportadas por las operaciones estándar.
    - Las clases IPeticionesDAO e IEquiposDAO son interfaces que declaran todas las operaciones que se van a poder realizar sobre los datos de tipo Petición y Equipo.
    - Las clases PeticionesDaoImpl y EquiposDaoImpl implementan los métodos declarados en las interfaces IPeticionesDAO e IEquiposDAO, utilizizando también métodos declarados en las interfaces IPeticionesJPA e IEquiposJPA.

- Servicio.

Implementa la lógica del negocio utilizando las clases de las capas anteriores, a través de la interfaz IPeticionesService, IEquiposService e IBlockchainService. Sus métodos se implementan en la clase PeticionesMantenimientoServiceImpl, EquiposServiceImpl y BlockchainServiceImpl respectivamente.

- Controlador.

Las clases PeticionController, EquiposController y BlockchainController definen y exponen como recursos REST, las rutas necesarias para realizar las operaciones declaradas en las clases de la capa del Servicio. Obtiene información de la base de datos, la modifica o añade. Es el punto de acceso de la API REST.

Así la clase PeticionesController expone las operaciones declaradas en la interface IPeticionesService y la clase EquiposController las declaradas en la interface IEquiposService.

La clase BlockchainController expone las operaciones declaradas en la interface IBlockchainService. Se utiliza para el acceso a la API que FireFly expone para el acceso a la información que la red blockchain contiene. También utiliza operaciones de la clase IPeticionesService para almacenar la información obtenida de la red.

- Modelo.
  La clase Peticion o Equipo conceptualiza los datos necesarios para procesar una petición de mantenimiento, o bien un equipo respectivamente, y que están almacenados en la base de datos. Mapea en objetos java los datos almacenados en la base de datos y sus relaciones, definiendo las operaciones que se pueden realizar con ellos.

1.8.1.1. API REST.

El uso del módulo Spring Web facilita al backend exponer una serie de operaciones a través de los endpoints definidos en la capa del controlador:

- GET.
  - /peticiones
    Entrega todas las peticiones de mantenimiento almacenadas.
  - /peticiones/{id}
    Entrega una petición de mantenimiento cuyo identificador (id_Peticion) se pasa por parámetro.
  - /peticiones/numeropeticion/{numeropeticion}
    Entrega una petición de mantenimiento cuyo identificador se pasa por parámetro.
  - /equipos
    Entrega todos los equipos almacenados.
  - /equipos/{id}
    Entrega un equipo cuyo identificador (id_Peticion) se pasa por parámetro.
  - /equipos/referencia/{referencia}
    Entrega un equipo cuya referencia se pasa por parámetro.
  - /equipos/numserie/{numeroserie}
    Entrega un equipo cuyo número de serie se pasa por parámetro.
  - /equipos/anyadir/{idequ}/{idpet}
    Asigna un equipo a una petición de mantenimiento que se pasan por parámetros.
  - /blockchain/obtenerEquipo
    Obtiene de la red blockchain información de un equipo cuyo id se pasan en el body de la petición como parámetro. Se usa para actualizar el estado de mantenimiento de un equipo en el centro reparador, por ejemplo.
  - /blockchain/buscarEquipoPorNumeroSerie
    Obtiene de la red blockchain información de un equipo cuyo numero de serie se pasanen el body de la petición como parámetro. Se usa para actualizar información registrada del equipo .

- POST.
    - /peticiones
      Añade a la base de datos la información contenida en una petición de mantenimiento que se pasa como una cadena JSON en el body de la petición al servidor.
    - /equipos
      Añade a la base de datos la información contenida en un equipo que se pasa como una cadena JSON en el body de la petición al servidor.
    - /blockchain/crearEquipo
      Añade a la blockchain la información contenida en un equipo que se pasa como una cadena JSON en el body de la petición al servidor.

- PUT.
    - /peticiones
      Actualiza la base de datos con la información contenida en una petición de mantenimiento que se pasa como una cadena JSON en el body de la petición al servidor.
    - /equipos
      Actualiza la base de datos con la información de un equipo que se pasa como una cadena JSON en el body de la petición al servidor.
    - /blockchain/actualizarEquipo
      Actualiza la información de un equipo existente en la red, con la petición de mantenimiento y la que fecha que se envía a un centro reparador. Esto ocurre cuando se asigna la fecha de envío.
    - /blockchain/actualizarEstadoMantenimiento
      Actualiza el estado de mantenimiento de un equipo.
    - /blockchain/actualizarEstadoFinal
      Actualiza el estado de mantenimiento de un equipo.

- DELETE.
    - /peticiones/{id}
      Elimina de la base de datos la petición de mantenimiento cuyo identificador (id_Peticion) se pasa como parámetro.
    - /equipos/{id}
      Elimina de la base de datos la información de un equipo cuyo identificador (id_Equipo) se pasa como parámetro.

1.8.1.2. Base de datos.

La base de datos se ha implementado con MySQL 8.0 según el modelo E-R definido en el apartado 1.7.4.1. Se ha construido la base de datos a través de un script SQL almacenado en el fichero peticionesequiposdb.sql. Además, se ha introducido una serie de peticiones y equipos para realizar pruebas a través de otro script SQL insertpeticionesequipos.sql. Ambos ficheros acompañan esta memoria.

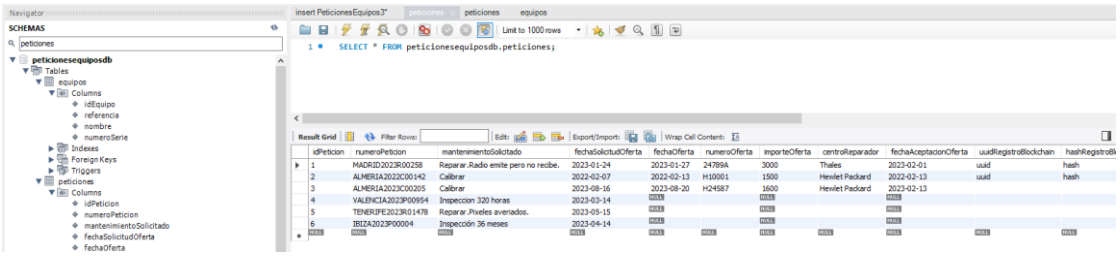

Figura 47: Mysql Workbench mostrando tablas, campos y datos de prueba, (fuente: elaboración propia).

Para conectar la base de datos al entorno de programación se deben de seguir dos pasos:

- Añadir la fuente de datos al entorno de programación.
  A través del menú View/Tools Windows/Database se despliega un panel que permite añadir nuevas fuentes de datos:

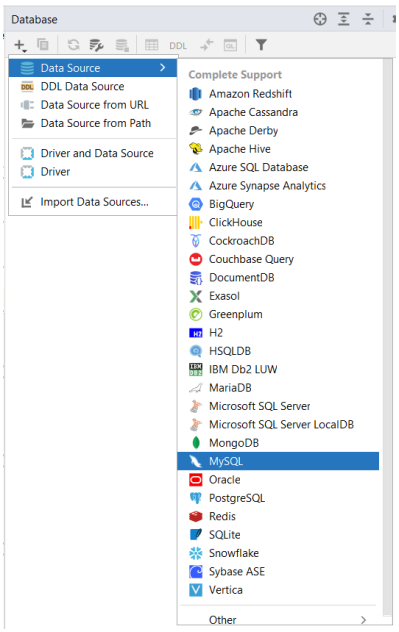

Figura 48: menú View/Tools Windows/Database, (fuente: elaboración propia).

Ya sólo queda completar la información para que la base de datos esté conectada al entorno de programación:

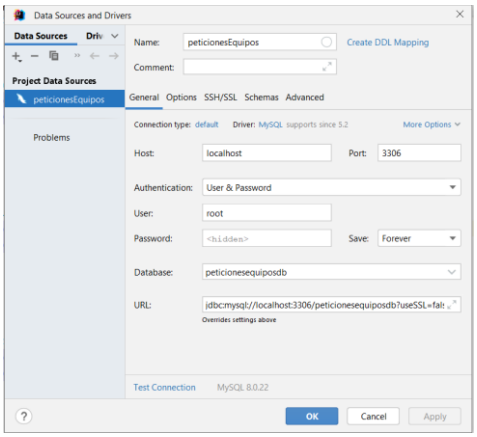

Figura 49: Formulario para definir la conexión a la base datos, (fuente: elaboración propia).

- El siguiente paso es conectar el origen de datos al proyecto que estamos desarrollando. Para ello se define la capa de persistencia a través del menú View/Tools Windows /Persistence, que abre un formulario para asignar la capa de persistencia del proyecto al origen de datos que acabamos de definir:

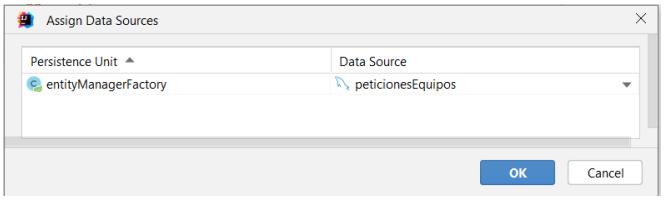

Figura 50: Selección de Data Source, (fuente: elaboración propia).

Una vez implementada la base de datos, conectada al entorno y al proyecto, este accede a los datos a través del módulo Spring Data JPA. Como se describió en el aparatado 1.8.1.1, se añadió al proyecto al configurar mediante https://start.spring.io los módulos de Spring que se iban a utilizar.

1.8.2. Desarrollo del cliente web.

Para implementarlo se ha desarrollado en primer lugar la interfaz de usuario, que dispone de una página principal, un página para crear peticiones de mantenimiento nuevas, una página para buscar peticiones ya iniciadas y una página para consultar el estado de un equipo enviado a mantener al centro reparador. Completa el cliente una página para listar los equipos del inventario y otra para editar su información o visualizarla.

En segundo lugar, se ha desarrollado la lógica que solicita datos al backend o la red blockchain a través de sus API's.

1.8.2.1. Implementación del cliente web.

Para la implementación del cliente web se ha desarrollado un microservicio que utiliza varios módulos del framework Spring:

- Spring Boot DevTools para que reinicie de forma automática la aplicación cuando se realizan modificaciones en el código.
- Spring Web para implementar el patrón MVC y que la aplicación utilice las operaciones del API RESTful que el backend expone.
- Spring Data JPA con el objetivo de utilizar sus funciones para gestionar los listados de peticiones y equipos.

- Thymeleaf como motor de plantillas html para implementar las paginas web de la aplicación.

Para ello se ha utilizado Spring Initializr al que se accede desde la web https://start.spring.io, cuya configuración se puede observer en la siguientes imagen:

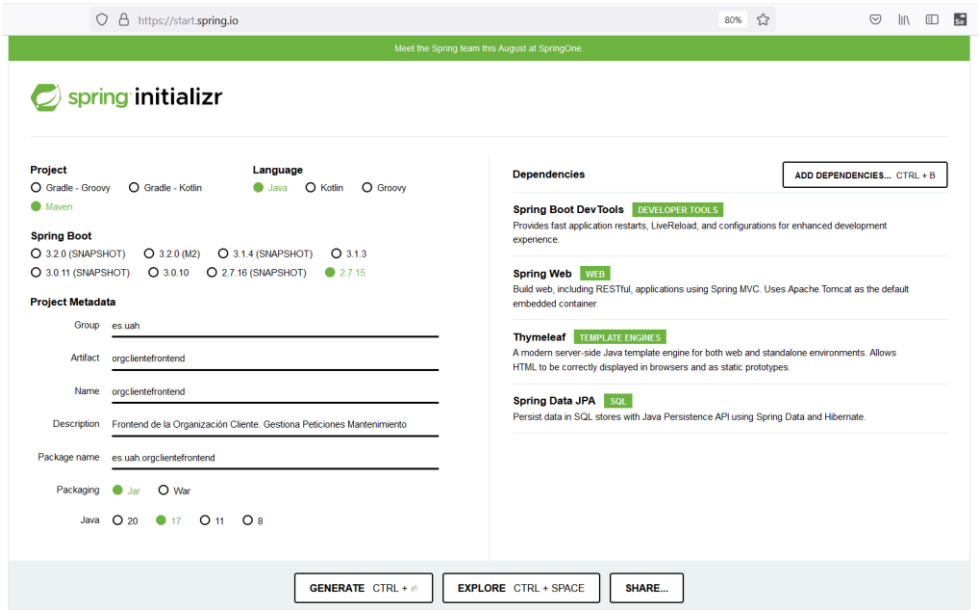

Figura 51: Selección de componentes microservicio frontend, (fuente: elaboración propia).

Al pulsar el botón Generate, se descarga el código que servirá de base para el desarrollo del frontend. Incluye las dependencias necesarias y las configuraciones básicas para comenzar el desarrollo del microservicio.

Se ha utilizado el entorno de programación IntelliJ IDEA 2023.2.1 Ultimate Edition y jdk 17.

- Descripción de las clases.

El cliente web dispone de una capa que repite los servicios expuestos por el backend (API Rest definida en el apartado 1.8.1.2) para poner utilizarlos.

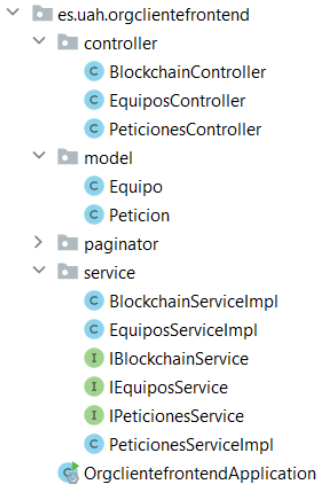

Figura 52: Capas del cliente web, (fuente: elaboración propia).

- Bean Equipo y Peticion.
  Representan los objetos equipo y petición del microservicio backend. Almacenan las operaciones y propiedades de dichos objetos.
- Bean RestTemplate.
  Para que la comunicación entre los microservicios de backend y frontend sea posible se añade un bean de RestTemplate OrgclientefrontendApplication.
- Clases del controlador
  Las clases Equipos Controller y PeticionesController tienen la responsabilidad de utilizar los métodos REST que el microservicio de backend expone. La clase BlockchainController es la encargada de utilizar la API que FireFly expone para el uso de la red blockchain. Utilizan las operaciones declaradas en las interfaces IEquiposService, IPeticionesServices e IBlockchainService.
- Clases del servicio.
  Las interfaces agrupadas en el paquete service IEquiposService, IPeticionesServices e IBlockchainService que declaran las operaciones utilizadas por las clases del controlador, tienen implementados los métodos que declaran en las clases EquiposServiceImpl, PeticionesServicesImpl e BlockchainServiceImpl respectivamente.

- Clases Lógica de Presentación.
  Para la capa de presentación se ha utilizado el motor de plantillas HTML Thymeleaf. Dentro del directorio del proyecto resources se pueden encontrar las siguientes páginas web:
  - home.html.
    Página principal muestra los enlaces para crear una nueva petición de mantenimiento, generar los listados de peticiones, buscar peticiones o gestionar el inventario de equipos.
  - listPeticion.html y LisEquipo.html.
    Son las páginas que muestran listados de peticiones o equipos, además disponen de botones para añadir o borrar del listado alguno de sus elementos.
  - formPeticion.html y formEquipo.html.
    Se utilizan para añadir o editar peticiones o equipos respectivamente.
  - mostrarPeticion.html.
    Visualizan la información de una petición.
  - buscarPeticion.html.
    Esta página se utiliza para introducir la referencia el número de serie o de petición de mantenimiento por los que se quiere buscar una petición.

- Clases de Paginación.
  - Agrupadas en el paquete paginator paginator-nav.html pageItem.html y pageRender.html se encargan de gestionar la paginación de los listados.

1.8.2.2. Interfaz de usuario.

- home.html
  Página principal con tres enlaces hacia el resto de páginas.

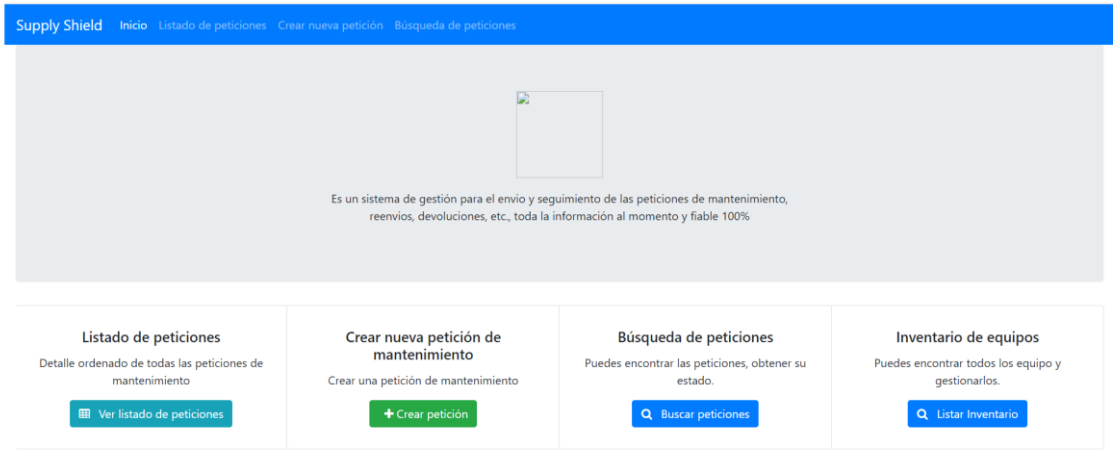

Figura 53: Imagen página home.html, (fuente: elaboración propia).

- formPeticion.html
  Esta página se utiliza para añadir una nueva petición de mantenimiento o para editarla

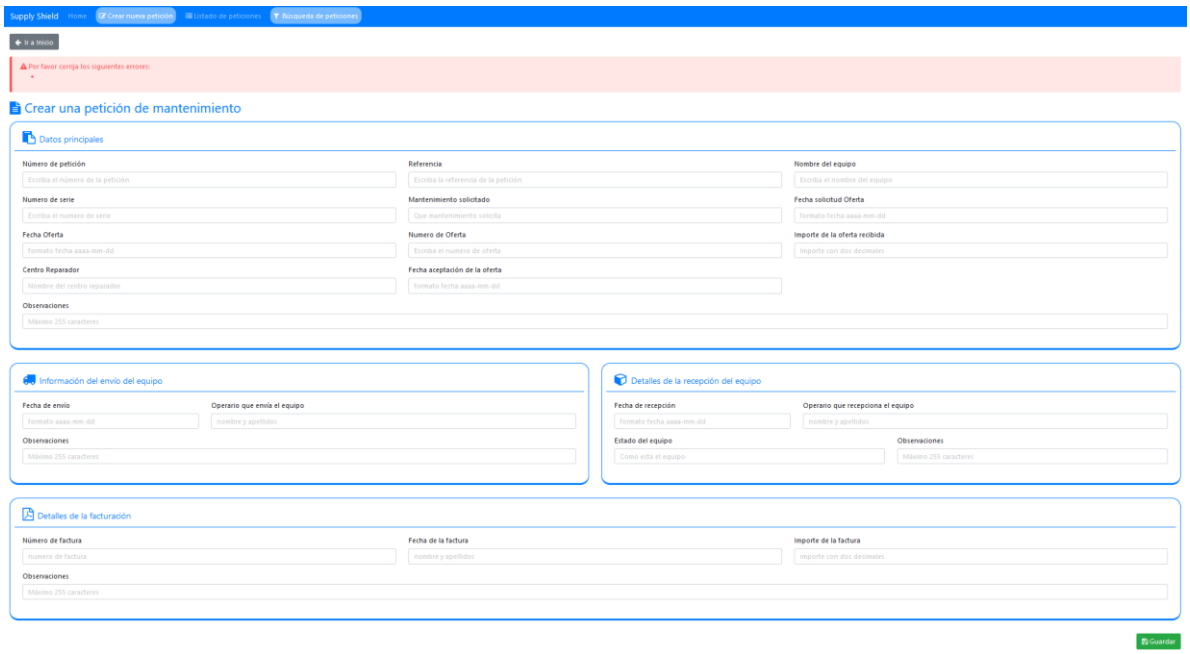

Figura 54: Imagen página formPeticion.html (fuente: elaboración propia).

- listPeticion.html
  Muestra un listado de peticiones de mantenimiento que pueden ser editadas o borradas con los botones situados a la derecha.

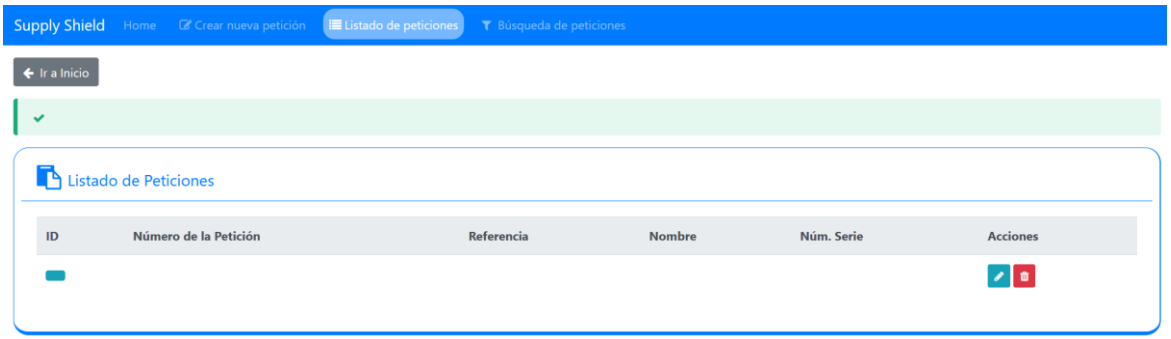

Figura 55: Imagen página listPeticion.html (fuente: elaboración propia).

- formBuscarPeticion.html
  Sencilla página de búsqueda de peticiones de mantenimiento por número de serie, referencia y número de petición.

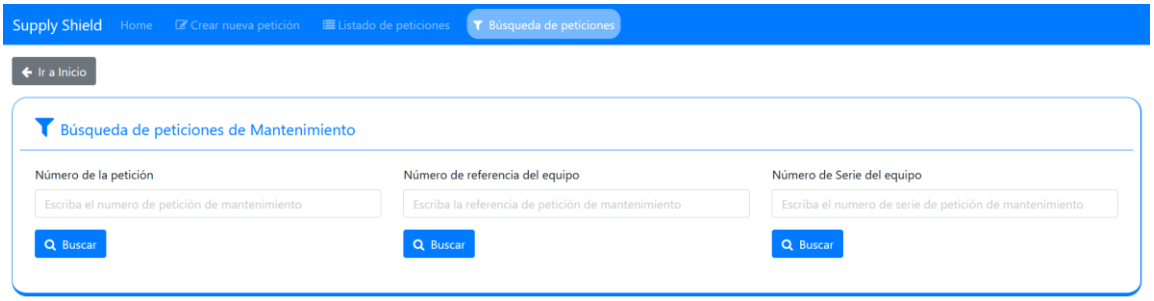

Figura 56: Imagen página formBuscarPeticion.html (fuente: elaboración propia).

- formEquipo.html.
  Esta página se utiliza para añadir un nuevo equipo o para editarlo.

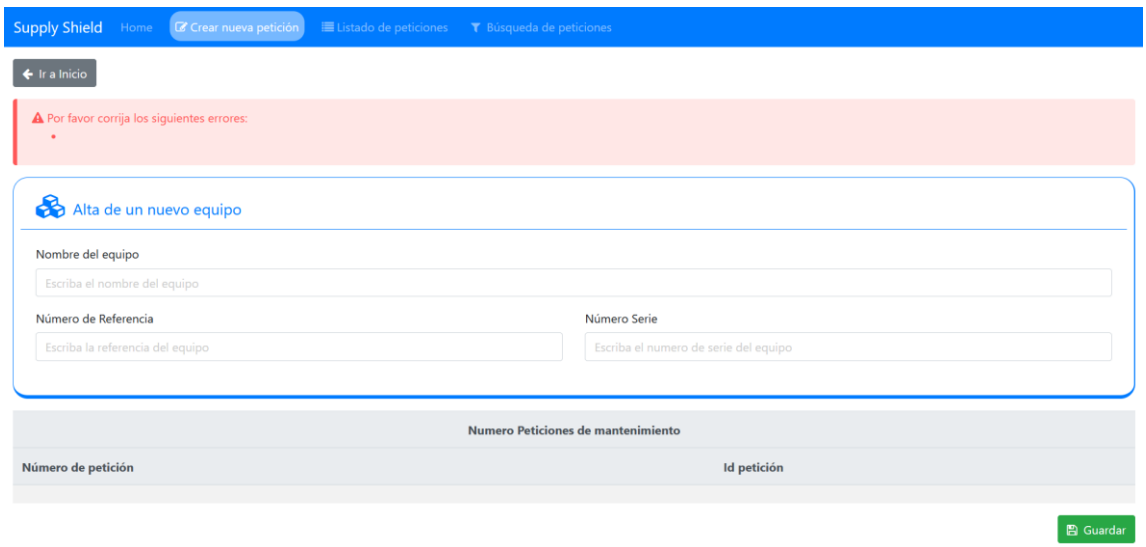

Figura 57: Imagen página formEquipo.html (fuente: elaboración propia).

- lisEquipo.html.
  Muestra un listado de equipos que pueden ser editados, eliminados o añadidos a una petición de mantenimiento con los botones situados a la derecha.

Figura 58: Imagen página listEquipo.html (fuente: elaboración propia).

- Matriz Caso uso – funcionalidad – Interfaz de usuario.

A continuación, se puede observar la matriz que relacciona el caso de uso con la interfaz que permite realizarlo. Ante cualquier modificación del software permite conocer los ficheros afectados y estimar mejor el coste de recursos y tiempo necesarios.

Caso de uso	Organización	Actor	Funcionalidad	Interfaz de usuario
CU-1	Cliente	Control de producción	añadir peticiones	formPeticionesMantenimiento.html
CU-2	Cliente	Control de producción	buscar peticiones	formBuscarPeticiones.html
CU-3	Cliente	Control de producción	actualizar peticiones	formPeticionesMantenimiento.html
CU-4	Cliente	Control de producción	enviar equipos	formPeticionesMantenimiento.html
CU-5	Cliente	Control de producción	recibir equipos	formPeticionesMantenimiento.html
CU-6	Cliente	Operario	enviar equipos	formPeticionesMantenimiento.html
CU-7	Cliente	Operario	recibir equipos	formPeticionesMantenimiento.html
CU-8	Centro Reparador	Operador	consulta equipos	formBuscarPeticiones.html
CU-9	Centro Reparador	Operador	registrar tarea mantenimiento	formPeticionesMantenimiento.html
CU-10	Cliente	Control de producción	Listar peticiones mantenimiento	listPeticionesMantenimiento.html
CU-11	Cliente	Control de producción	Listar inventario de equipos	listEquipo.html
CU-12	Cliente	Control de producción	añadir/modificar equipos	formEquipo.html
CU-13	Cliente	Control de producción	seleccionar peticion/equipo	listEquipo.html

Figura 59: Matriz caso de uso-actor-funcionalidad-interfaz de usuario (fuente: elaboración propia).

1.8.3. Desarrollo de la red Blockchain. Estado actual para implementar redes blockchain.

Para el desarrollo de una red blockchain que permita interconectar varias organizaciones empresariales involucradas en el uso de una aplicación común, cada miembro de la red debe incorporar a sus propia colección de aplicaciones, una nueva infraestructura que les permita acceder a la red e interoperar con el resto de miembros.

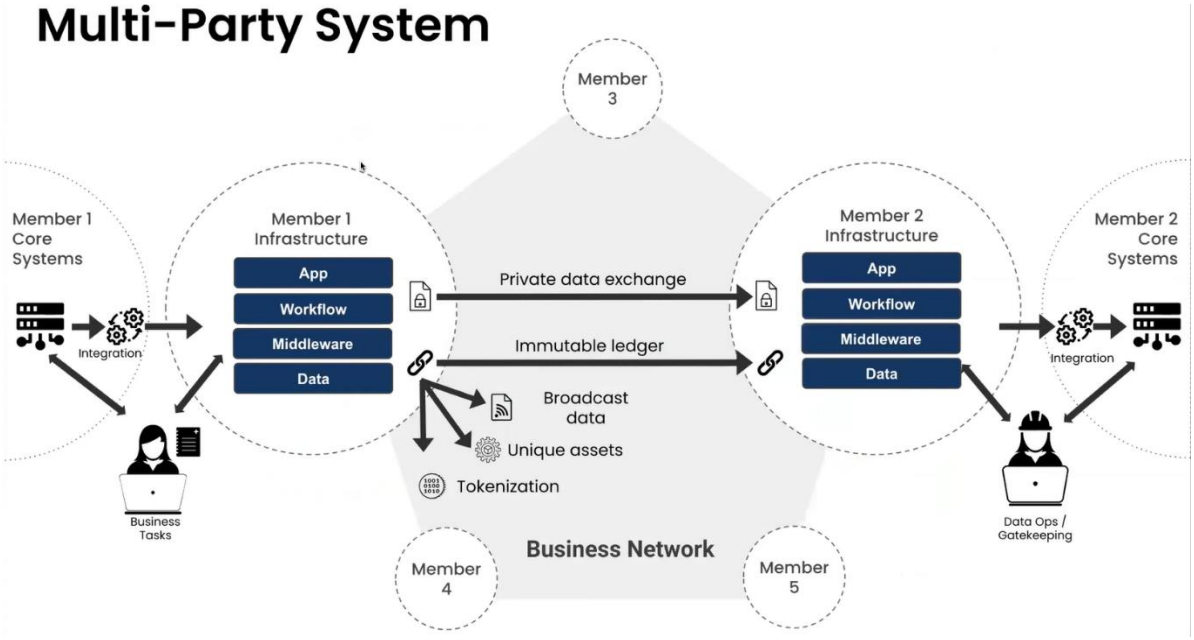

Figura 60: Red Blockchain Genérica (HyperLedger, 2023e)

Se puede observar en la figura 60, una red genérica blockchain privada en la que sus miembros se conectan a la red, intercambian información de forma pública o privada, tienen su propia base de datos y su interfaz de aplicación. Toda la información que es privada está aislada del resto de miembros de la red, pero todos deben acordar el modelo de datos para que pueden sincronizarlos uniformemente en sus sistemas informáticos.

Parece sencillo, pero es necesario intercambios de datos y documentos, capacidad para enviar mensajes de forma privada, secuenciación de eventos, gestionar la identidad de los miembros o gestion de tokens, por ejemplo. Además hay que tener en cuenta que la mayoría de los datos estarán situados fuera de la red por razones de privacidad o de tamaño.

Se puede pensar que los pasos para el desarrollo de una aplicación así podrían ser:

1. Escribir un smart contract con la lógica de la aplicación.
2. Desplegar un nodo blockchain.
3. Desarrollar la aplicación web común a todos los miembros.
4. Averiguar como implementarlo.

A priori, una estimación del tiempo que podría llevar podría ser del orden de 4-6 meses (Kaleido, 2022a). Sin embargo una vez comenzado realmente el desarrollo, se descubre que el proyecto es diferente:

1. Es necesario diseñar como utilizar blockchain.

2. Es necesario construir toda la infraestructura necesaria para la red blockchain.
3. Es necesario desarrollar una API para el acceso a la blockchain desde la aplicación web.
4. El proyecto en realidad está mucho más lejos de desplegarse de lo estimado.

La realidad es que un proyecto de este tipo puede llevar desarrollarlo entre 24-48 meses. (Kaleido, 2022a)

En un inicio el proyecto utilizaba una red blockchain como parte de una aplicación web, y la realidad lo ha convertido en una parte muy importante, en un desarrollo de la infraestructura necesaria.

El apartado 1.8.3.4 describe de forma muy resumida como se realizaría un despliegue sobre AWS de una blockckain construida utilizando Hyperledeger Fabric.

Lo que se busca para la implementación del presente proyecto es:

1. Definir los activos a almacenar en la blockchain y un modelo de datos,
2. Describir los procesos y los eventos producidos en los que los miembros de la red se ven envueltos para orquestarlos.
3. Escribir la API que le permite a la aplicación web utilizar la red blockchain.
4. Desplegarlo todo de forma sencilla.

Para eliminar toda la complejidad descrita común a las aplicaciones que utilizan una red blockckain y simplificar su uso Hyperledger ha desarrollado Hyperledger FireFly.

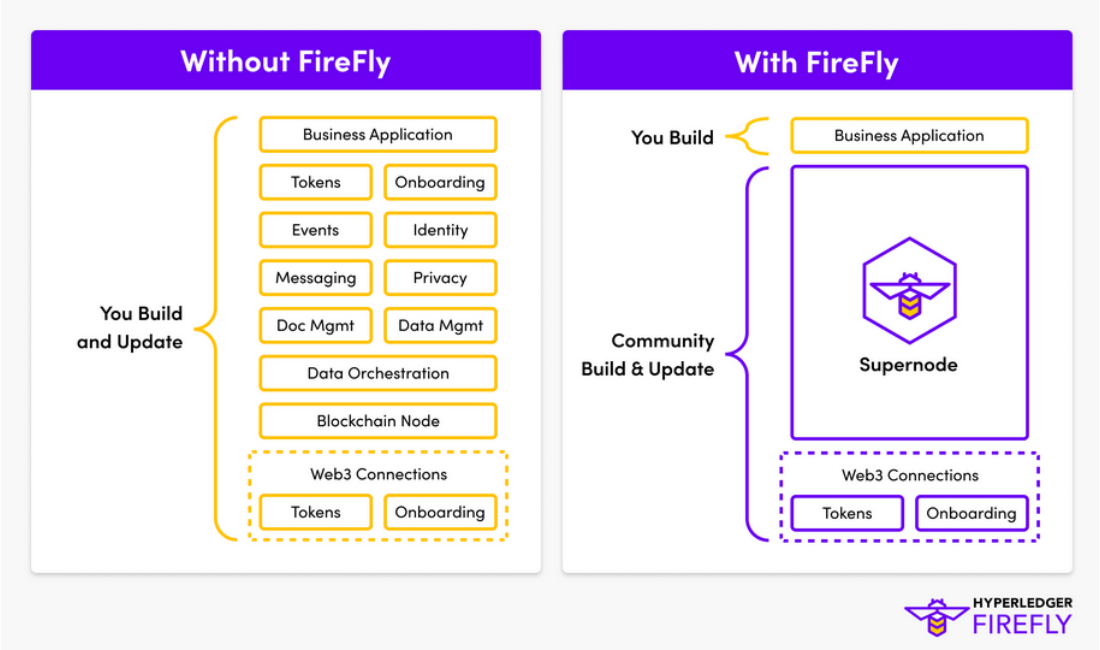

Figura 61: Funciones Supernodo FireFly (Hyperledger, 2023d)

1.8.3.1. Modelo de Blockchain con FireFly.

En la figura 5 se puede observar todas las funcionalidades necesarias que han sido encapsuladas en un nodo FireFly. Teniendo esto en cuenta la red blockchain genérica de la figura 55 se simplifica enormemente (Hyperledger, Huperledger Foundation, 2023b):

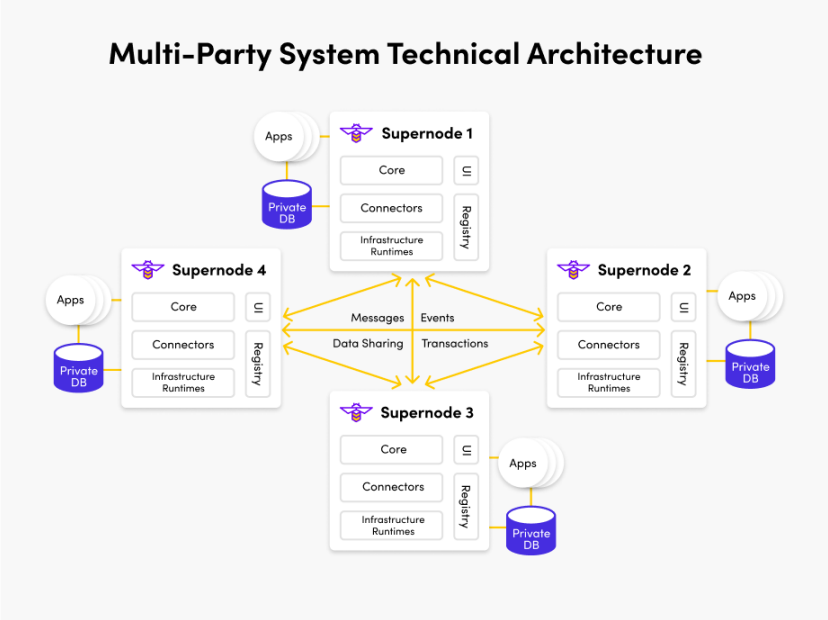

Figura 62: Red Blockchain Genérica Firefly (Hyperledger, 2023b)

Esto permite a los desarrolladores concentrar sus esfuerzos en la lógica del negocio que están implementado, y utilizar las funcionalidades que Firefly les ofrece para interactuar con el resto de miembros de la red.

Únicamente es necesario escribir un smart contract que define las funciones de la red blockchain, y FireFly generará automáticamente una API para que sea consumida por las aplicaciones web que se está desarrollando. Se puede observar un resumen esquemático del proceso en la figura 63.

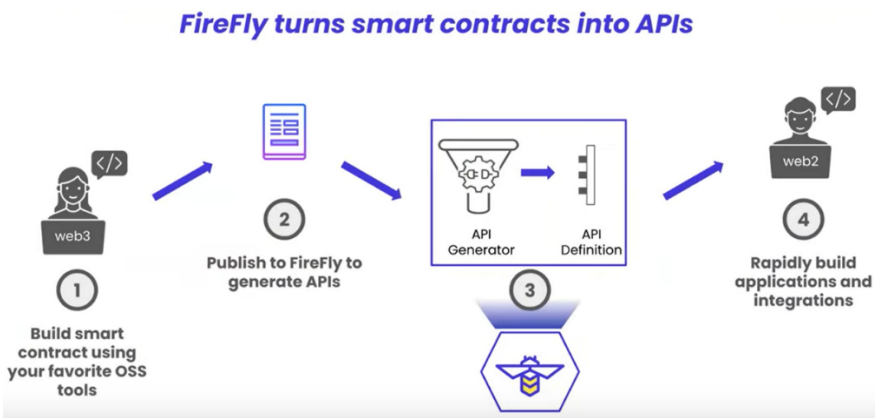

Figura 63: Cómo construir aplicaciones web 3 rápidamente (Kaleido, 2022b).

La descripción de la API generada puede consultarse en la dirección:

https://hyperledger.github.io/firefly/v1.2.0/swagger/swagger.html

1.8.3.2. Arquitectura solución basada en FireFly.

Se describen a continuación la forma de compartir información tanto pública como privada y las herramientas necesarias para comunicarse con la blockchain.

- **Sistema de almacenamiento y consulta de información.**

Cada miembro de la red blockchain dispone en FireFly de un nodo IPFS (Interplanetary File System) que almacena la información, y está conectado al resto de nodos IPFS de la red, a través de conexiones peer to peer (IPFS, 2023). Esta forma de almacenar los datos fuera de la red blockchain en nodos IPFS permite almacenar grandes cantidades de información.

Los nodos IPFS proporcionan una API que ofrece funciones para almacenar o consultar información. Así cada vez que se almacena un archivo proporciona un hash que puede ser utilizado posteriormente para acceder a dicho archivo. Al estar conectados entre sí los nodos IPFS pueden comunicarse entre sí.

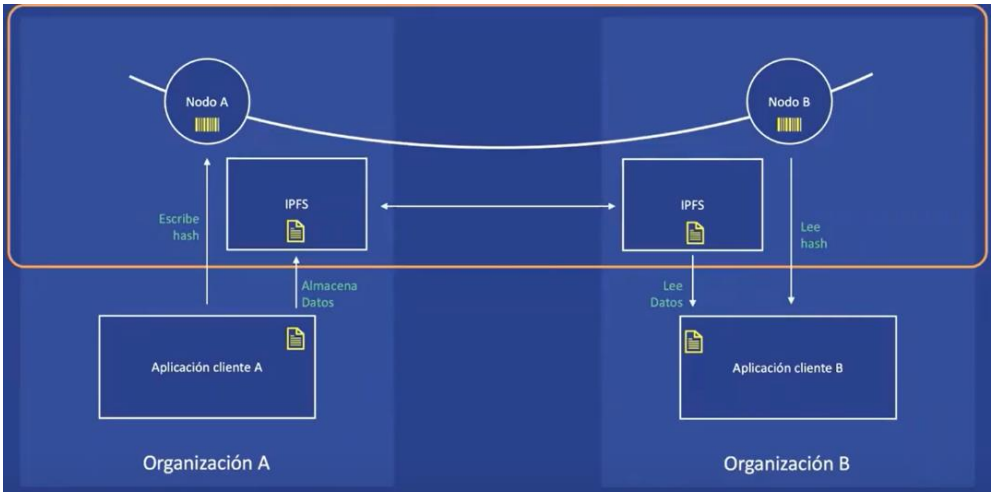

Figura 64. Comunicación pública de información fuera de la red blockchain(Kaleido, 2022c).

Datos off-chain públicos. Comunicación a través de Nodo IPFS.

A modo de ejemplo, se puede observar en el esquema de la figura 64 cómo la Aplicación Cliente A almacena información en el sistema IPFS de su nodo y recibe un hash que lo identifica. Dicho hash es escrito en el nodo A de la red blockchain a través de un smart contract (o de la función de la API generada por Firefly prevista para ello). El hash se va a propagar por la red blockchain y el nodo B lo va a recibir. A partir de ese momento el hash puede ser leído por la Aplicación B y lo puede utilizar para consultar el archivo correspondiente, proporcionándolo a su nodo IPFS y recibiendo a cambio el archivo. Dicho archivo se recibe por el sistema IPFS a través de la conexión peer to peer que mantienen entre sí todos los nodos IPFS.

Este sistema se utiliza cuando se trata de información que es compartida por todos los miembros de una red.

En el caso de que sea información privada sólo entre algunos miembros se garantiza la privacidad a través de otro tipo de nodo: Document Exchange.

Datos off-chain privados. Comunicación a través de Nodo Document Exchange.

En el caso de que la comunicación sea privada, únicamente entre algunos miembros de la red, se utiliza como sistema seguro de archivos un nodo Document Exchage (Kaleido, Document Exchange, 2023). En este caso no existe una comunicación peer to peer, sino un registro con la identidad de los diferentes nodos Document Exchange (On Chain Registry).

Cada miembro de la red tiene un nodo Document Exchange y dispone de una clave pública y una clave privada. La clave privada está almacenada en el nodo Document Exchange y sólo el miembro dueño de la clave tiene acceso a ella. Sin embargo, la clave pública está almacenada en el registro al que todos los miembros de la red tienen acceso: On Chain Registry.

Aunque la identificación se realiza a través de On Chain Registry, si la Organización A quiere enviar información a la Organización B se realiza a través de un canal de información como puede ser Kafka.

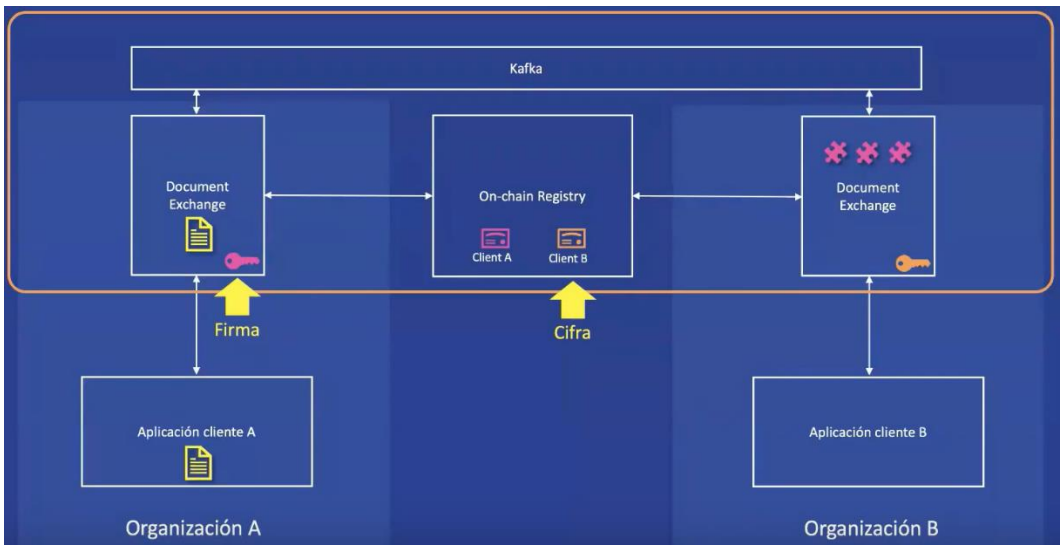

Figura 65: Comunicación privada de información (Kaleido, 2022c).

El esquema de la figura 65 ilustra cómo la Organización A quiere enviar un archivo con información a la Organización B, y para ello almacena dicho archivo en su nodo Document Exchange, que lo va a firmar con la clave privada de la Organización A y cifrarlo con la clave privada de la Organización B, destino de la información. El archivo viaja a través de Kafka hasta el nodo Document Exchange de la organización B, donde es descifrado con la clave privada de la Organización B y utiliza la clave pública de la Organización A para verificar la identidad de la Organización que envió la información.

Es decir, el sistema no sólo asegura que únicamente la organización destino pueda leer la información, sino que identifica sin lugar a dudas el emisor de la información.

Figura 66. Emisión recibo de la información (Kaleido, 2022c).

Una vez el archivo es descifrado y el emisor autenticado, el sistema utiliza la clave privada de la organización destino para firmar un recibo y la clave pública de la organización origen para cifrarlo como se puede observar en el esquema de la figura 61.

El recibo cifrado viaja por Kafka hacia la Organización A, que lo descifra con su clave privada y verifica la firma con el certificado público de la organiza B. Una vez hecho esto la Organización A puede leer el recibo.

En resumen, una vez finalizado todo este proceso, la organización A envió un documento y tiene un recibo que asegura que la Organización B lo recibió, y ahora la organización B tiene una copia de dicho documento.

- **EthConnect.**

Software open source que transforma llamadas API Rest en transacciones blockchain. Es el responsable en FireFly de interpretar el código de un smart contract y convertir todos los métodos que encuentre en llamadas restful de una API REST. De esta forma una aplicación puede utilizar la API Rest generada para interactuar con la blockchain y se encapsula la complejidad de todos los detalles necesarios para realizarlo. (Github Hyperledger FireFly Ethconnect, 2023)

- **EvenStrems.**

La función de esta herramienta open source es convertir eventos de la red blockchain en llamadas REST (webhooks) o bien en Push Notifications (websockets). El objetivo es que la red blockchain realice llamadas REST a nuestra aplicación cuando ocurre un evento como puede ser una nueva transacción, por ejemplo.

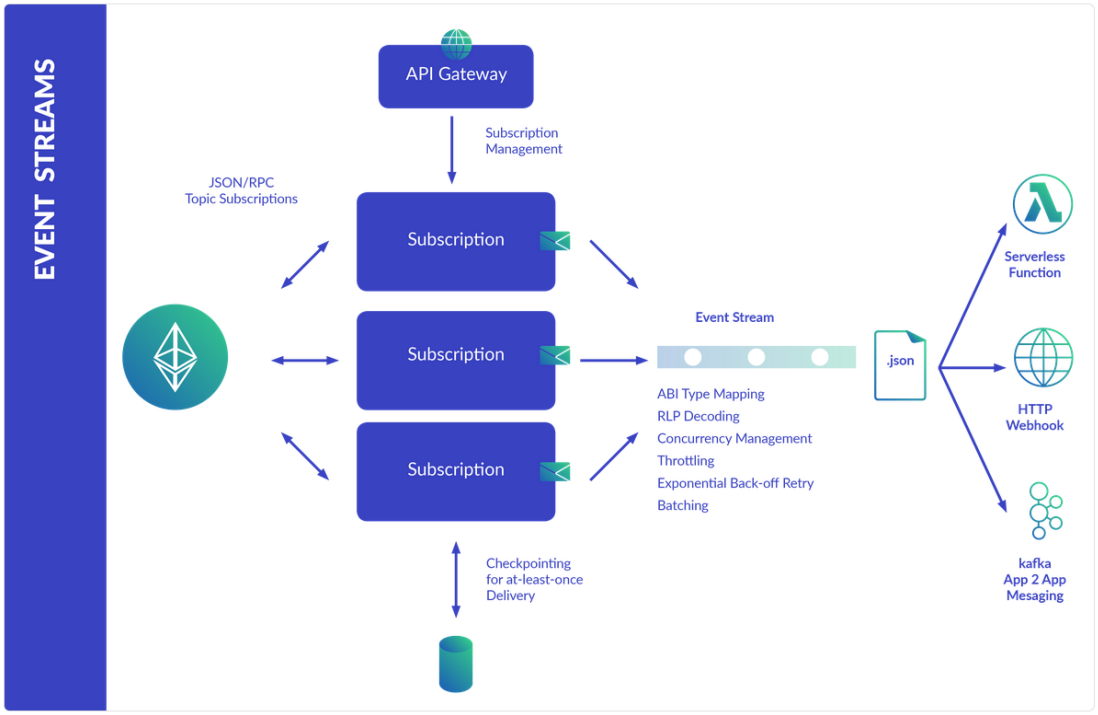

Figura 67. Esquema funcional Event Stream (Kaleido Documentacion Desarrolladores, 2023).

- Arquitectura solución basada en FireFly.

En la figura 68 se puede observar el esquema de una solución que utiliza FireFly para conectar la aplicación de una organización a una red blockchain.

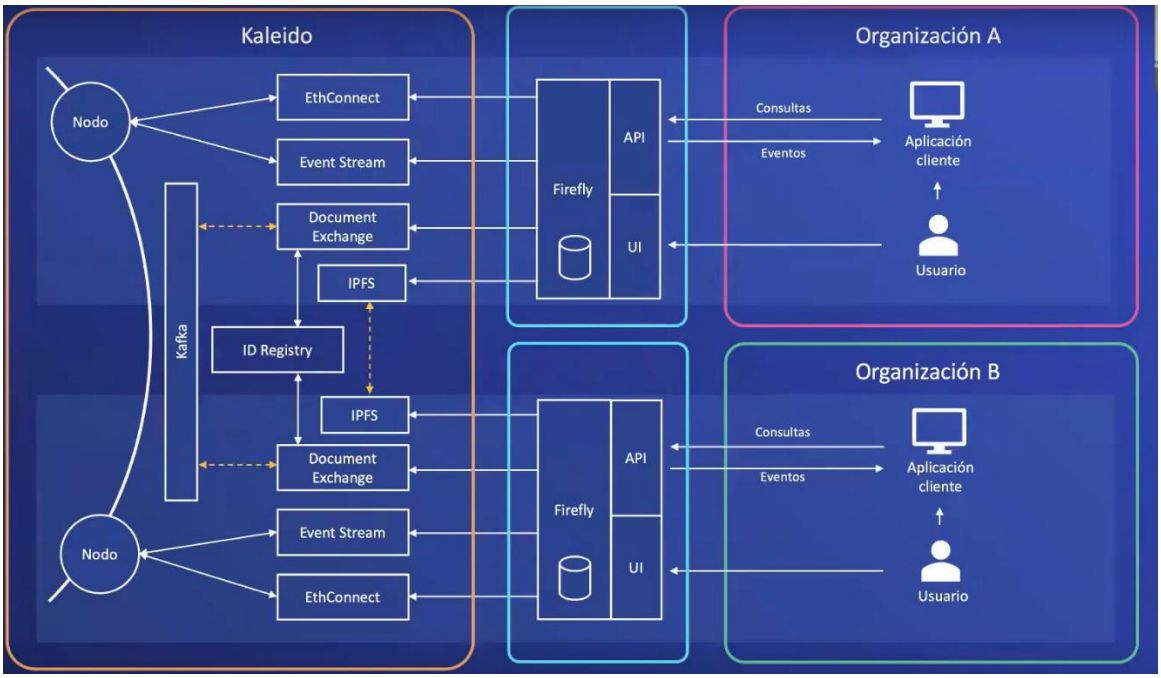

Figura 68. Esquema solución utilizando Firefly (Kaleido, 2022c).

Las aplicaciones de las organizaciones A y B (Dapp, aplicaciones distribuidas) realizan llamadas API a EthConnect, y van a ser transformarlas en transacciones blockchain que se ejecutan en sus respectivos nodos. De la misma forma cada vez que se produce un evento en la red blockchain se transmite hacia las aplicaciones utilizando Event Stream.

La aplicación de cada organización puede almacenar información a través de llamadas a su nodo IPFS si los datos pueden ser públicos, o a través de llamadas a su nodo Document Exchange si los datos a transmitir son privados con algún miembro específico de la red. Para esto utiliza la autenticación de identidad a través del On-chain Registry y Kafka para la transmisión de la información.

FireFly encapsula toda la complejidad, sincroniza y coordina todas las llamadas a EthConnect, Event Stream, IPFS y Document Exchange. El usuario puede interactuar bien con la interfaz que FireFly ofrece o bien a través de la interfaz de la aplicación distribuida. Esta dapp ejecuta llamadas a la API o alguno de sus procesos internos interactúa con la API a petición del usuario.

Además, cada herramienta que se conecta a FireFly son plug-in. No es obligatorio su uso y cada desarrollador puede utilizar la herramienta que crea mejor para su caso de uso, siempre que cumpla los requisitos de un plug-in para FireFly.

FireFly ha sido desarrollada por la compañía Kaleido y donada para open source a la Fundación Hyperledger.

1.8.3.3. Modelo Blockchain con FireFly en entorno local de desarrollo.

FireFly está además disponible para desarrollar localmente aplicaciones e incluye herramientas y ejemplos. Permite disponer de más de una instancia representativas de varias organizaciones, que interaccionan entre ellas a través de la aplicación que se está desarrollando, y las herramientas necesarias para controlar los eventos que están ocurriendo. Tiene la ventaja de no depender de un servicio on-line del cual no siempre se conoce con precisión su estado.

Para ello se utilizan Docker Container para cada parte del sistema con las mismas herramientas, sustitutos o simulaciones que permiten que el funcionamiento sea el mismo en local que on-line.

Las distintas partes del sistema se pueden ver en la figura 69:

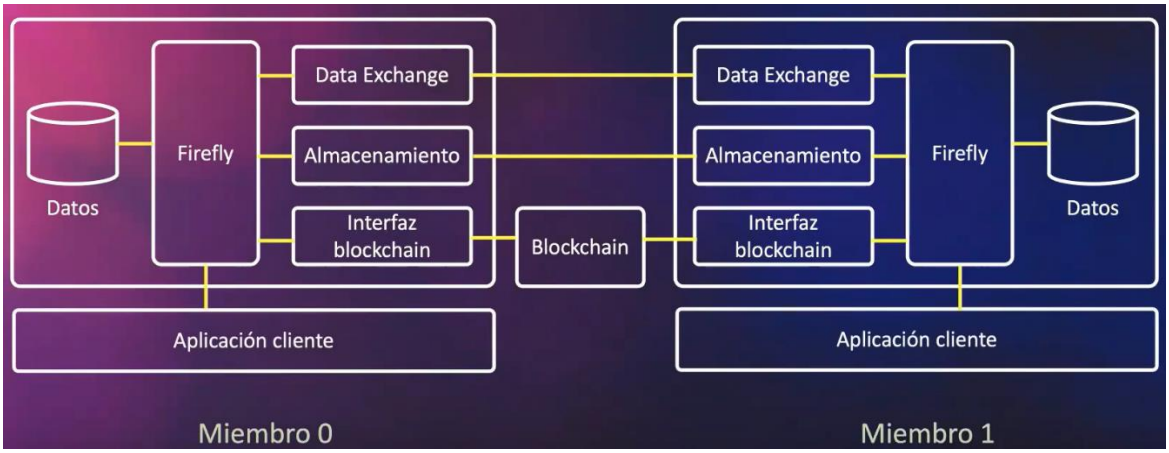

Figura 69. Parte de FireFly Local (Kaleido, 2022c).

Cada parte representa una herramienta en la versión local de FireFly.

Para la parte de Datos se ha utilizado PostGreSQL. El almacenamiento un sistema IPFS y el interfaz blockchain eth connect, ambos como en la versión on-line. La parte del intercambio privado de datos se utiliza DX HTTPS (Data Exchange https), un sistema de transferencia seguro (Hyperledger, Hyperledger FireFly, 2023c) utilizando https en vez de Kafka, y la blockchain se despliega utilizando Geth (authors, 2023)

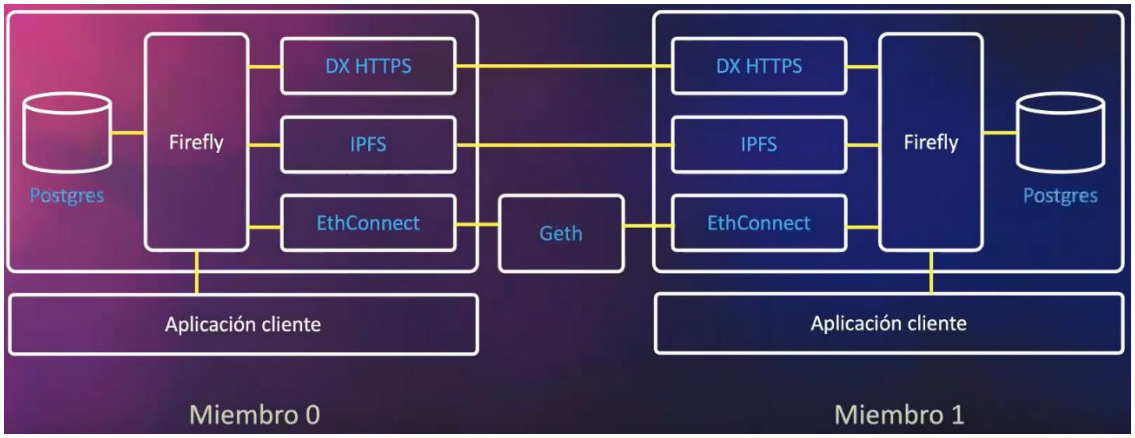

Figura 70. Herramientas utilizadas en FireFly Local (Kaleido, 2022c).

Cuando se realiza un despliegue, dispone además de una interfaz web para cada miembro de la red generado y poder observar lo que ocurre.

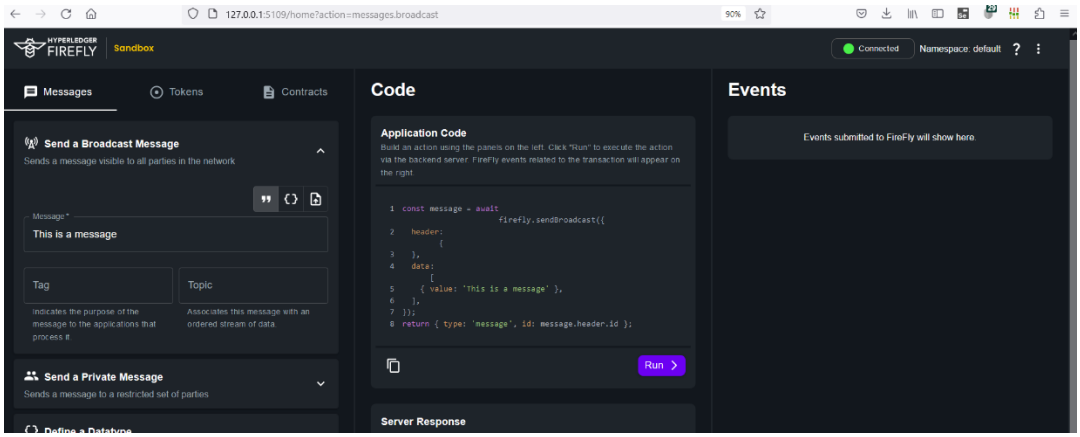

Figura 71. Interfaz de un miembro de la red (fuente: elaboración propia).

Para gestionar todo esto FireFly dispone de una consola de comandos que lo permite, FireFly CLI. La instalación es muy sencilla y sus instrucciones originales se encuentran en https://hyperledger.github.io/firefly/gettingstarted/ . En cualquier caso, en el apartado 1.8.3.5 Implementación con Hyperledger FireFly, se explica la instalación realizada para este proyecto.

1.8.3.4. Blockchain con Fabric sobre AWS.

Para el desarrollo de la red blockchain, otra opción es utilizar el framework Hyperledger Fabric y las herramientas que Amazon Web Services (AWS) pone a disposición del usuario para dimensionarla, gestionarla y desplegarla. La red blockchain debe ser capaz de mantener actualizadas las vicisitudes de los

equipos enviados a mantener al centro reparador una vez abandonan la organización cliente. Para ello almacena, actualiza y registra el estado de dichos equipos.

El desarrollo de la red se puede divididir en dos partes. Por un lado, desplegar una red blockchain que almacene los equipos enviados a mantener junto a un smart contract que los gestione. Por otro lado, es necesario añadir a la red, un servidor conectado a ella para cada miembro de la red que exponga al exterior un API REST tradicional. Así es posible que los miembros de la red puedan interactuar con la blockchain para gestionar los equipos enviados a mantener.

Se ha intentado implementar esta opción sobre AWS, pero la complejidad del proceso de construcción de la infraestructura necesaria, precisa de un conocimiento profundo de AWS para completarla y solo se ha conseguido parcialmente. La ventaja de esta herramienta es el completo control que el desarrollador tiene sobre la red construida. La desventaja es la necesidad de mantenerla actualizada.

El resto del apartado explica de forma resumida, los pasos seguidos para la construcción de la infraestructura de la red para dos miembros, uno el centro reparador y otro la organización cliente.

A continuación se puede observar la complejidad de la arquitectura construida para una red con un solo miembros:

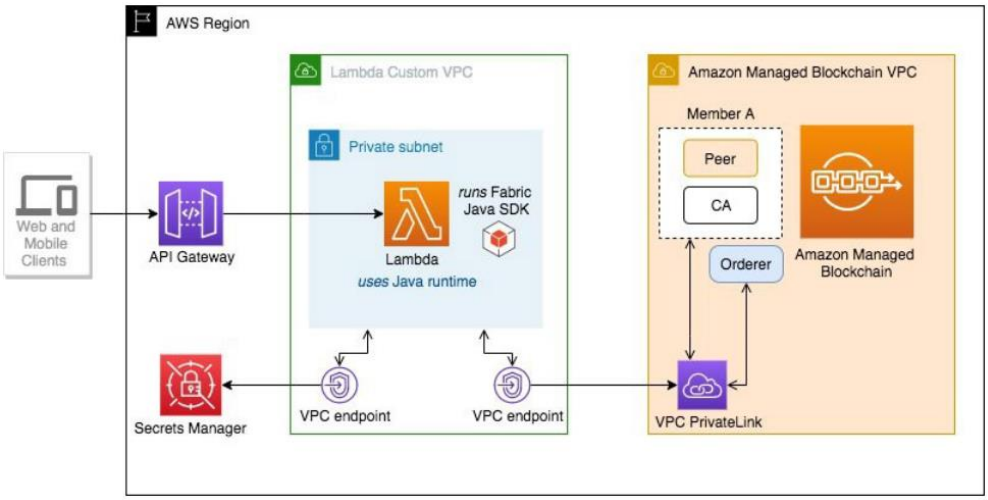

Figura 72. Arquitectura aplicación web sobre AWS Blockchain (Adhikari, 2020) .

La arquitectura completa implementada con AWS se puede observar en la anterior ilustración (Adhikari, 2020) .

Para crear la red necesaria, AWS ofrece una consola de administración (Amazon Web Service, 2020a). Esto se realiza desplegando un VPC endpoint que conecta los sistemas del centro reparador y la organización cliente. Se puede escalar horizontalmente, es redundante y con una alta disponibilidad.

En el siguiente diagrama (Amazon Web Services, 2020b) se representan los componentes del VPC endpoint, Fabric Ordered Service, que asegura que las transacciones se realizan en un orden determinista y que su estado en todos los nodos de la red sea coherente.

Fabric Certificate Authority es el componente que realiza las funciones de autoridad de certificación que identifica los diferentes clientes y servidores para asegurar que tienen autorizado el acceso a la red.

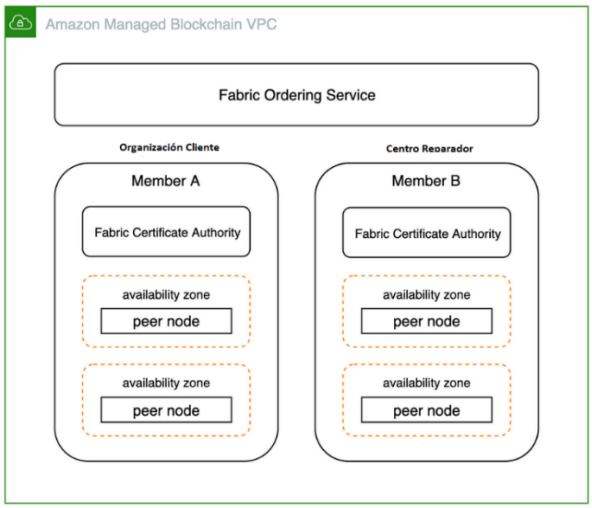

Figura 73. Esquema Fabric Ordering Service (Adhikari, 2020).

Para crear la red blockchain se siguen cuatro pasos:

- Selecciona el tipo de red, en este caso privada con Hyperledger Fabric y creación de un vpc para cada miembro de la red (Amazon Web Services, 2020c).
  Cada VPC endpoint tiene dos nodos situados en zonas horarias diferentes, para que en el caso de que una zona no esté disponible, el sistema pueda funcionar con el segundo nodo. Puede observarse en la siguientes imagen los nodos creados para uno de los miembros de la red.

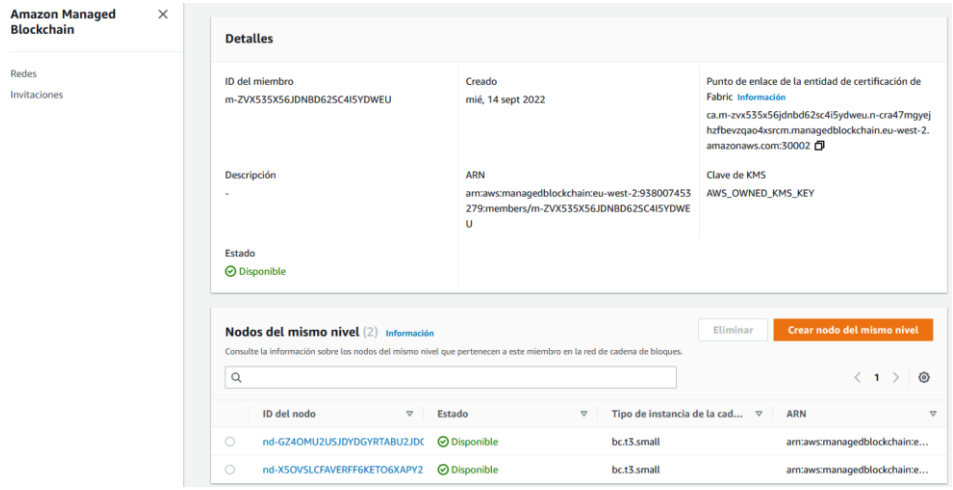

Figura 74. Nodos de uno de los miembros (fuente: elaboración propia)

- Invitar a los miembros de la red, en este caso la organización cliente invita al centro reparador a unirse.
  Para ello se ha utilizado una herramienta, aws cli, que permite introducir comandos para realizar actividades de gestión de la red blockchain, a través de la línea de commandos.

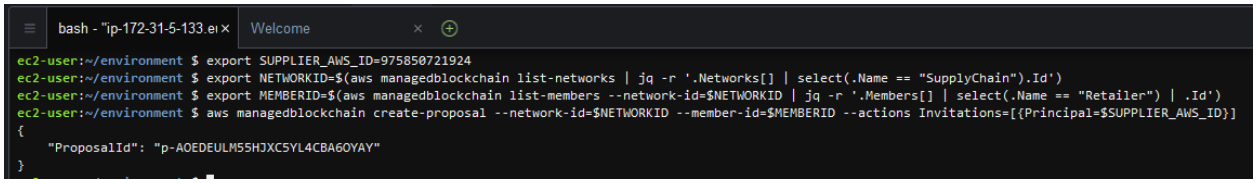

Figura 75. Invitación de uno de los miembros a unirse a la red (fuente: elaboración propia).

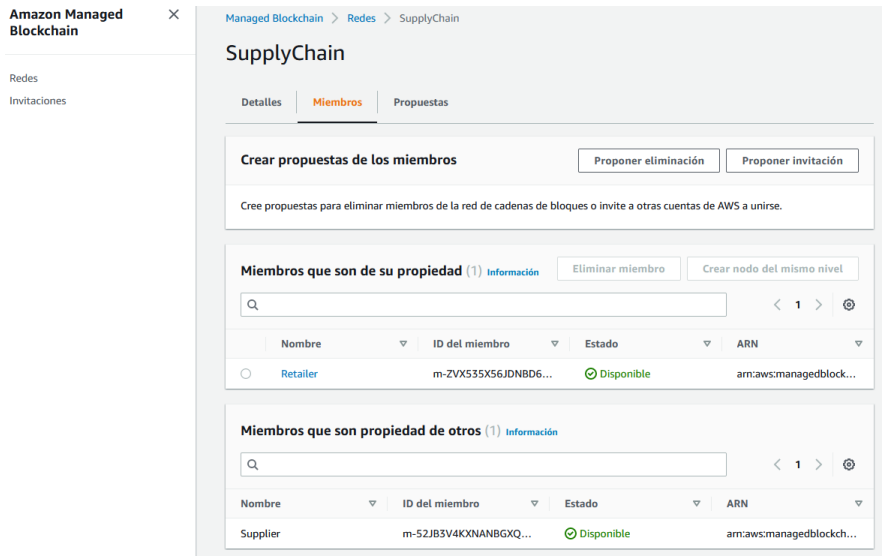

Figura 76. Vista Consola AWS con la red del cliente y el proveedor creada, (fuente: elaboración propia)

- Seleccionar los nodos que van a almacenar una copia de las transacciones.
  En este paso, se instala en cada miembro de la red, un cliente de Fabric a través del cual se realizan todas las interacciones con la red blockchain. Un diagrama de la arquitectura puede observarse en la siguientes imagen:

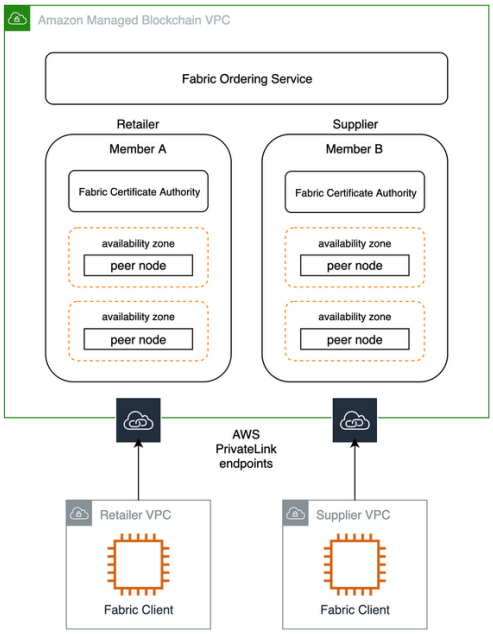

Figura 77. Diagrama arquitectura con los dos miembros de la red (Amazon Web Services, 2020d).

Este cliente Fabric se instala sobre una instancia ec2 (computador virtualizado ofrecido por AWS) (Amazon Web Service, 2022e), que se utilizará para implementar y configurar los contratos inteligentes de la aplicación que usa la blockchain.

- Implementación de las aplicaciones descentralizadas que van a realizar transacciones a través de nodos del mismo nivel.
  En este paso se desarrollan las aplicaciones gestionadoras de la red (smart contract) , que van a ser desplegadas en cada uno de los nodos a través de los clientes fabric instalados. Además se definen las identidades (control de producción en el caso de la organización cliente u operador en el caso del centro reparador que van a acceder a dichas aplicaciones).

  Para acceder a las funcionalidades que los smart contract ofrecen, son necesarias funciones lambda para encapsularlas y exponerlas al exterior a través de una API GraphQL. Para esto se utiliza la herramientas AWS AppLink.
  La arquitectura se puede observar en el siguientes diagrama (Amazon Web Services, 2020e):

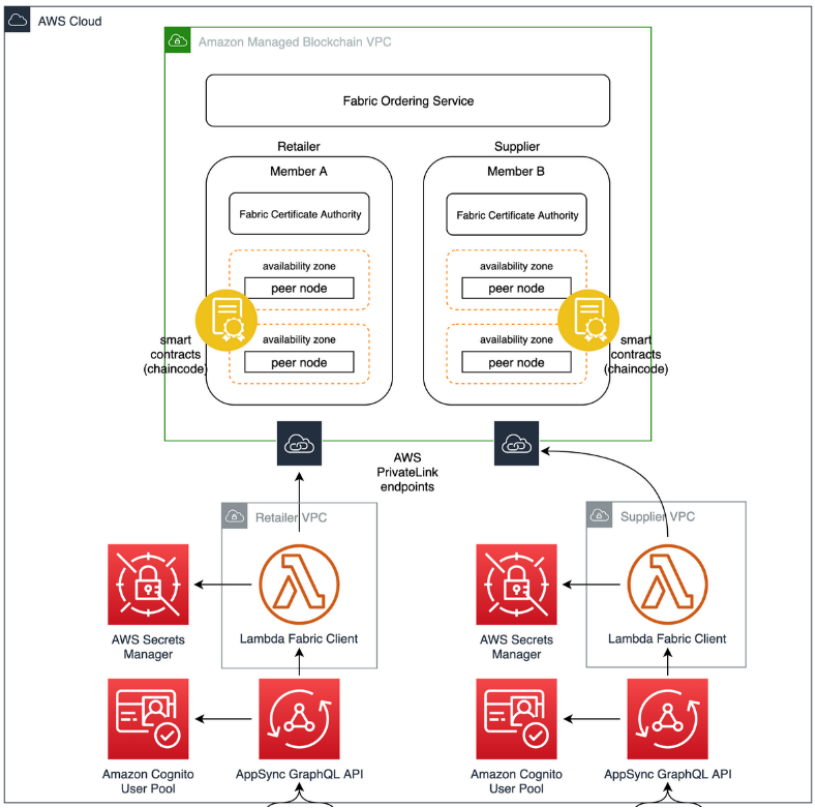

Figura 78. Arquitectura de acceso a la red de cada miembro diagrama (Amazon Web Services, 2020e)

- Funciones Lambda
  En AWS se define una única función, que se ejecuta de forma independiente, para cada una de las funciones definidas en el smart contract. En el caso de este proyecto el smart contract está definido en el punto 1.8.3.5 y se puede observer en la figura 84. Codifica funciones para crear, actualizar, eliminar o consultar información de los equipos que está situada en la red.
  En el caso de utilizar Hyperledger FireFly no es necesario realizarlo porque automáticamente genera esta capa de software al desplegar el contrato.

- GraphQL API.
  Una vez realizadas las funciones lambda, se utiliza la herramientas de AWS AppLink para generar una API GraphQL, que permita exponerlas y ser consumidas por un cliente web.
  Si se utilizase Hyperledger FireFly, genera una API REST a partir de la información contenida en un fichero con formato json, que se genera al compilar el smart contract, como se explica en la figura 86 del punto 1.8.3.5.

1.8.3.5. Implementación de la red Blockchain con Hyperledger Firefly Local.

- Instalación de Firefly.

Para la instalación de Hyperledeger Firefly, se ha seguido las instrucciones contenidas en https://hyperledger.github.io/firefly/gettingstarted/

Se realiza en tres pasos:

1.- Instalar FireFly CLI.

Para instalar la consola de comandos la computadora debe disponer de Docker, Docker Compose y Openssl. También se recomienda utilizar WSL2(Windows Subsystem for Linux 2).

En este caso la máquina de desarrollo tiene instalado WSL2 asi que es necesario bajar la última versión desde el enlace https://github.com/hyperledger/firefly-cli/releases/latest y ejecutar la máquina Linux el siguiente comando:

sudo tar -zxf ~/Downloads/firefly-cli_*.tar.gz -C /usr/local/bin ff && rm ~/Downloads/firefly-cli_*.tar.gz

La verificación de la instalación se puede realizar ejecutando el comando ff version.

2.- Configurar el entorno de desarrollo.

Para probarlo vamos clonar en la máquina de desarrollo el repositorio de ejemplos y utilizando el ejemplo data-transfer, ejecutar el comando ff init data-transfer 3. Esto configurará una red blockchain de tres miembros de nombre data-transfer. Instala en el directorio /home/nombre_usuario/.firefly/stacks/data-transfer/docker-compose.yml los archivos de configuración necesarios para que el entorno funcione.

3.- Ejecutar el entorno de desarrollo.

Para comprobar el funcionamiento correcto del entorno, con el comando ff start data-transfer, se ejecuta la red data transfer que se ha preparado previamente.

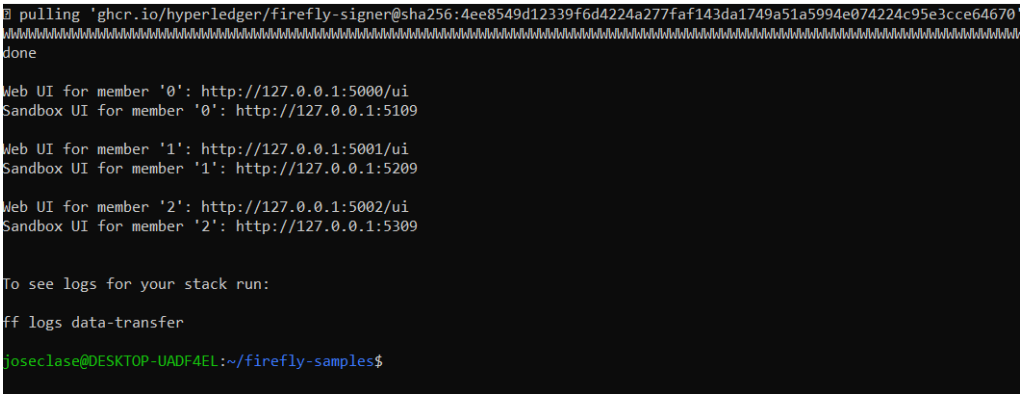

Figura 79. Ejemplo data-transfer arrancado mostrando direcciones de los miembros de la red (fuente elaboración propia).

En la figura anterior se puede observar que ya se puede acceder a los tres miembros creados, a través de la interfaz de usuario que para cada uno FireFly genera automáticamente. Para pararlo ff stop data-transfer.

- Ejecución de la red blockchain y su configuración.

1.- Arranque de la red.

Una vez arrancado el ordenador host con Linux WSL, se arranca el entorno WSL desde el menú de inicio/ Ubuntu on Windows:

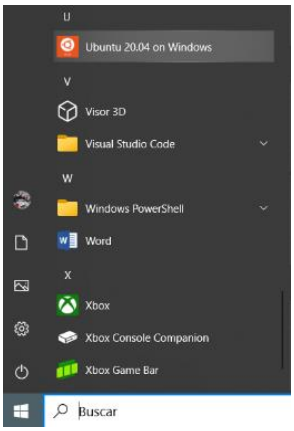

Figura 80. Arrancando Ubuntu con WSL (fuente elaboración propia).

Y en la consola que sea bre se arranca con el comando ff start data-transfer:

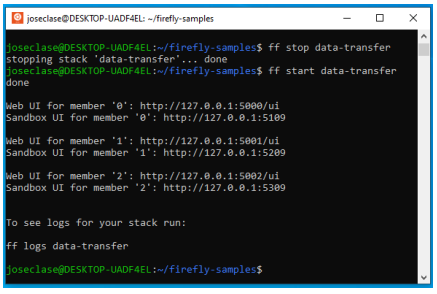

Figura 81. Consola Linux con WSL (fuente elaboración propia).

Si se abre el visualizador de docker se puede ver como el stack complete está en funcionamiento:

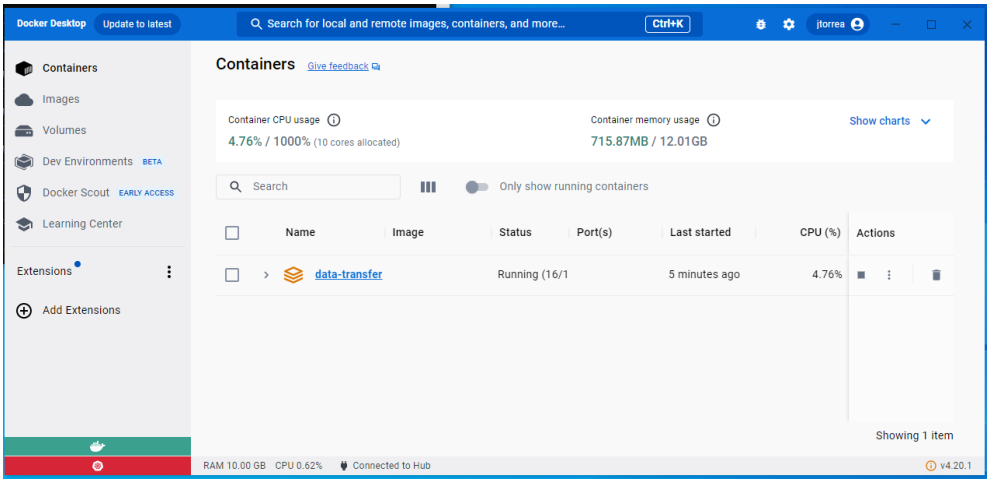

Figura 82: Interfaz gráfica Docker (fuente elaboración propia).

Y abriendo una ventada de navegador en las direcciones arriba indicadas se tiene acceso a la interfaz de los miembros de la red y sus sandbox como se puede ver:

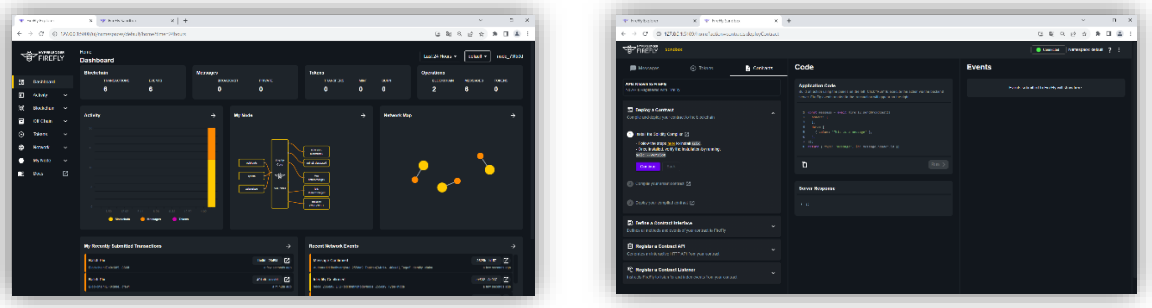

Figura 83: Interfaz gráfica de un miembros y su sandbox (fuente elaboración propia).

2.- Configuración: añadir smart contract, tipos de datos y generar API de la red.

Para añadir el smart contract es necesario disponer del compilador solc. Para instalarlo en WSL es necesario ejecutar tres comandos (Solidity, 2023):

- sudo add-apt-repository ppa:ethereum/Ethereum
- sudo apt-get update
- sudo apt-get install solc

Si se dispone del compilador únicamente es necesario seguir las instrucciones que el propio sandbox nos indica:

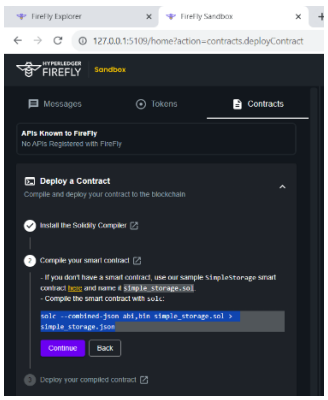

Figura 84: Comandos necesarios para compliar smart contract (fuente elaboración propia).

Desde el directorio donde está situado el código del Smart contract se ejecuta el comando de la figura 85. Una vez hecho esto se genera un fichero del mismo nombre con extensión json:

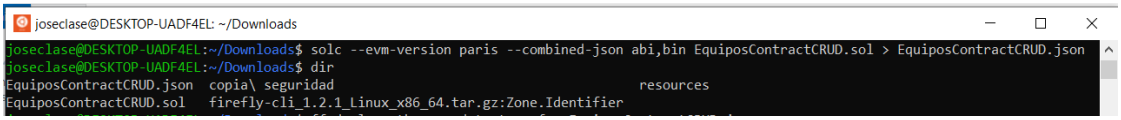

Figura 85: Comandos de compilación y resultado (fuente elaboración propia).

En este caso el fichero que se genera es EquiposContractCRUD.json. Se puede observar que se ha añadido la opción –evm-version paris al ejecutar el compilador. Esto es así porque, en el momento de elaborar este trabajo, la red privada de FireFly versión 1.21 funciona con la máquina virtual ethereum (EVM) parís. Hay que indicárselo al compilador solc, porque a partir de solc version 0.8.20 compila para la EVM shangai, que introduce un nuevo opcode. Al no soportarlo, cuando se intenta desplegar el contrato, genera el error "Error: http://127.0.0.1:5102 [500] {"error":"invalid opcode: PUSH0"} " (Andre, 2023).

Este fichero sirve para completar el paso 3: Deploy your compiled contract, que se observa en la figura 86. Para ello, como el propio sandbox indica, desde el directorio del fichero .json, se ejecuta el comando ff deploy <ethereum | fabric> <STACK_NAME> simple_storage.json .

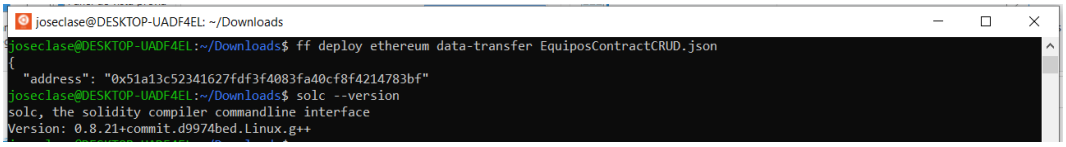

Figura 86: Comando de despliegue del contrato (fuente elaboración propia).

La consola, una vez termine el despliegue, devuelve la dirección del contrato en la red.

3.- Desarrollo del Smart contract.

El smart contract se ha desarrollado utilizando Remix IDE. El entorno se carga en el navegador e integra los elementos necesarios para editar el código, compilarlo, desplegarlo en una red de prueba y de forma automática genera un pequeño formulario para introducir o leer datos de la red y realizar pruebas.

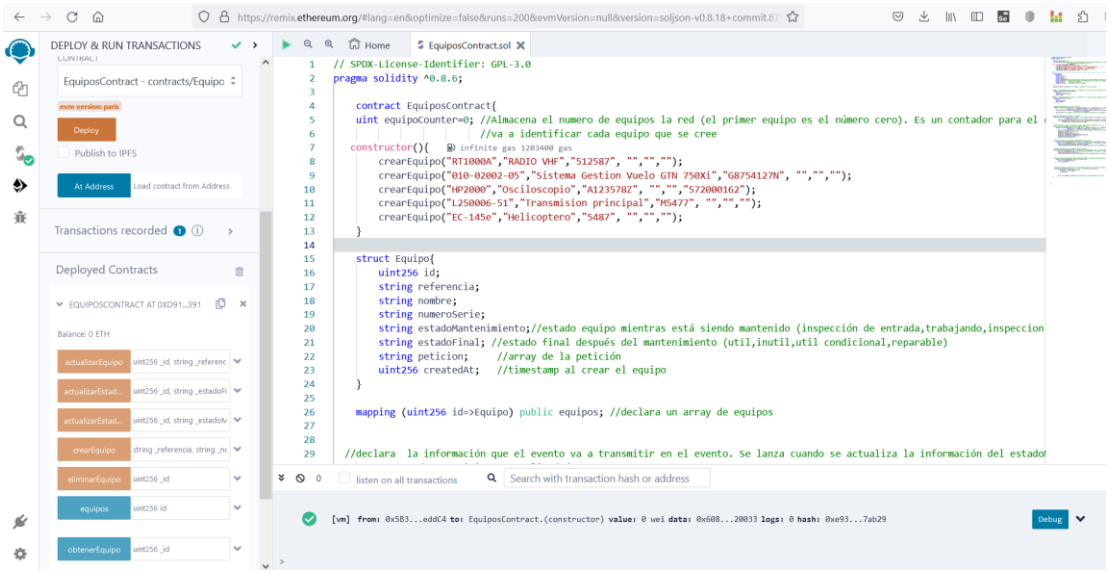

Figura 87: Comandos de compilación y resultado (fuente elaboración propia).

El ide Remix es sencillo de utilizar (Fazt, 2021). Una vez escrito el código, se graba el fichero con la combinación de teclas ctrl+s. Como se ve en la parte superior izquierda de la figura 81, aparecerá un círculo

verde con un check que significa sin errores o rojo si existe alguno. Si se pulsa en el botón deploy, el ide despliega el smart contract en una red de prueba y genera automáticamente un interfaz con el que se puede realizar pruebas de las funciones implementadas. Además, en la parte inferior dispone de una consola donde muestra información de las instrucciones que la red va ejecutando.

- Tipos de datos. Almacenamiento de la información.

El smart contract desarrollado utiliza un tipo de dato Equipo que almacena los datos de un equipo. Tomándolo como elemento, se declara una colección de equipos que almacena los datos de todos los equipos. Esta es la información que se almacenará en la red blockchain.

- Funciones.

Se han definido funciones para crear, consultar, actualizar y borrar equipos de la colección. En este punto hay que señalar que, aunque se elimine un equipo de la colección, la información de todas las transacciones realizadas permanece en la red blockchain. No se puede alterar ni eliminar.

- Eventos.

Se han definido dos eventos que se generan cuando se modifica el estado de mantenimiento de un equipo o se asigna su estado final, una vez realizado la tarea de mantenimiento solicitada. Estos eventos serán suscritos por el backend para almacenar la información transmitida cuando se produzcan.

Terminar de explicar el proceso de despligue en el sandbox y mostrar el api que genera de formaautomática donde se suscribe el cliente y el backend.

4.- Generación automática del API REST para la interacción con la red blockchain.

El siguiente paso es definir el API con la interfaz binaria que el compilador de solidity genera automáticamente durante el proceso de compilación, (está definido dentro del fichero json generado):

equiposContractCRUD.json

```
{
  "contracts": {
    "EquiposContractCRUD.sol:EquiposContract": {
      "abi": [
        {
          "anonymous": false,
          "inputs": [
            {
              "indexed": false,
              "internalType": "uint256",
              "name": "id",
              "type": "uint256"
            },
```

Figura 88: Definición ABI (fuente elaboración propia).

Y generar el interfaz ejecutando el método de FireFly GenerateContractInterface pulsado el botón Run del Sandbox:

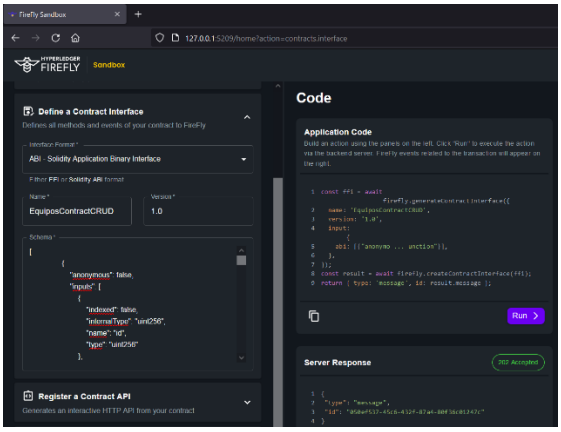

Figura 89: Definición y generación de API(fuente elaboración propia).

Se observa en al figura 89 cómo el servidor responde con un mensaje aceptando el contrato.

Y, por último, es necesario registrar el API con la dirección del smart contract ("0x51a13c52341627fdf3f4083fa40cf8f4214783bf"), completando los datos y pulsando en el botón Run de nuevo:

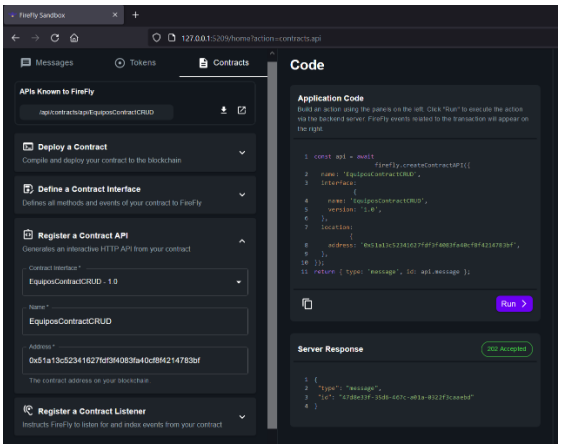

Figura 90: Registro del API con la dirección del smart contract (fuente elaboración propia).

Nuevamente se puede ver en la figura 90 una respuesta del servidor aceptando el registro, junto a un nuevo botón que se ha habilitado en la zona superior derecha: APIs Known toFireFly. Si se pulsa, se puede consultar el documento swagger generado automáticamente con la definición del API.

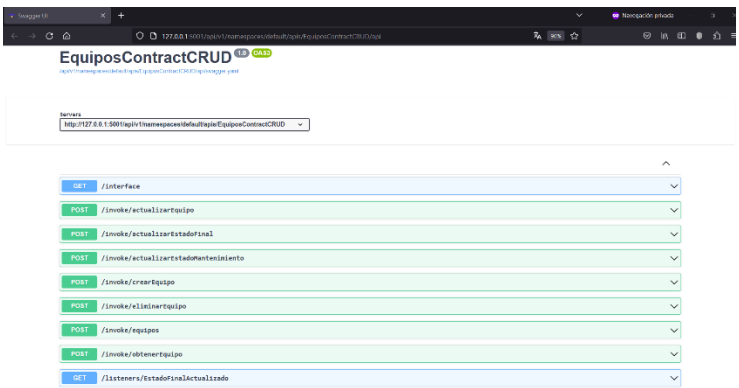

Figura 91: Swagger con la definición del API para interactuar con la red blockchain, (fuente elaboración propia).

Por último, el registro de los eventos en el apartado del Sandbox Register a Contract Listener se puede ver en la figura 92.

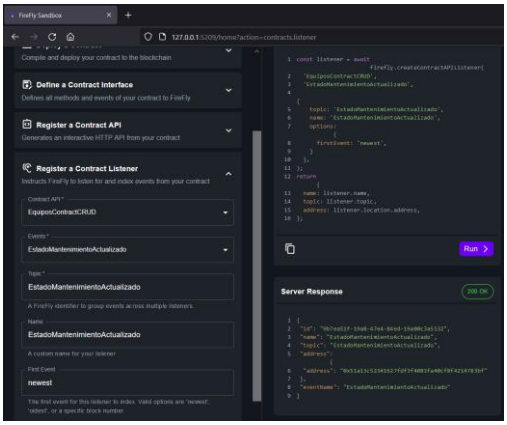

Figura 92: Registro de los eventos (fuente elaboración propia).

5.- Prueba del API REST para la interacción con la red blockchain.

Para probar el buen funcionamiento del método de despliegue tan sólo es necesario utilizar el api generado. Por ejemplo, se puede añadir un equipo como se muestra en la figura 93 utilizando /invoke/crearequipo:

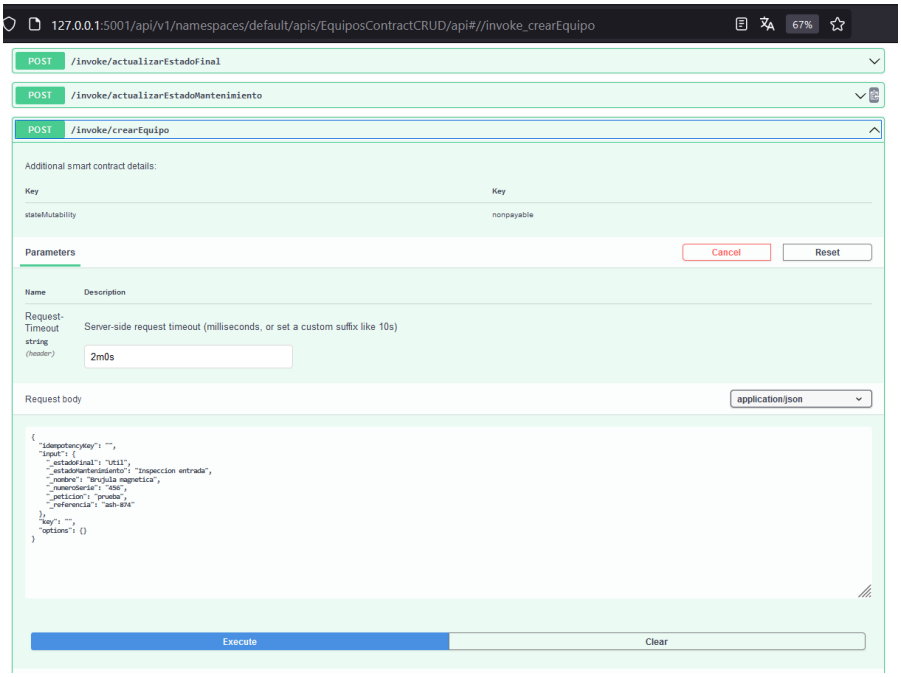

Figura 93: Petición para crear un equipo en la red blockchain (fuente elaboración propia).

Y utilizar el api /query/obtenerEquipo, en este caso sólo hay un equipo se pide el equipo 0:

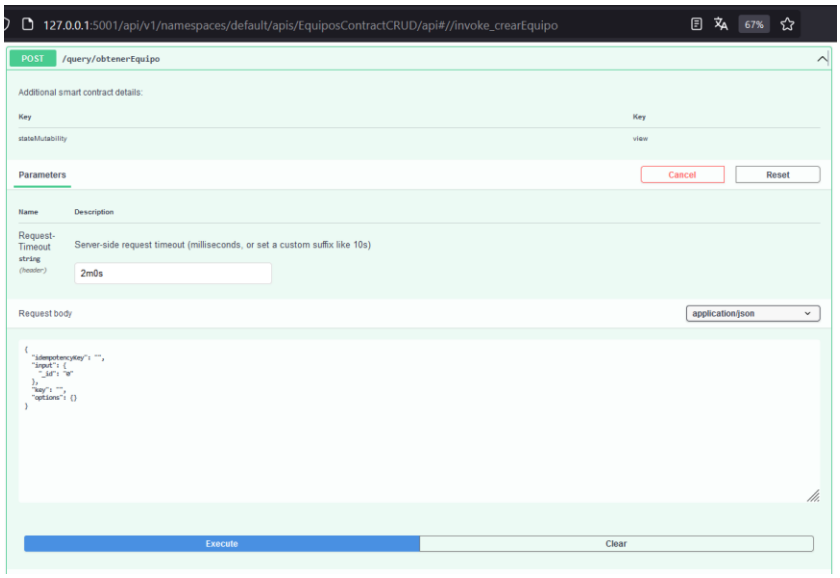

Figura 94: Petición para consultar el equipo creado (fuente elaboración propia).

Se obtiene como respuesta del servidor el equipo que se creó:

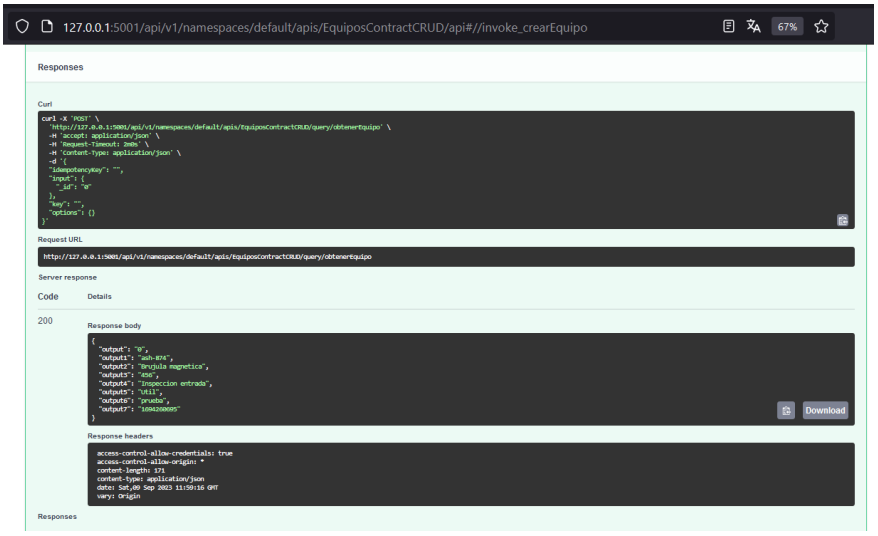

Figura 95: Respuesta del servidor con información equipo 0 (fuente elaboración propia).

Para integrar este API con el resto del proyecto tan sólo es necesario añadir una clase Controller más a los microservicios backend y frontend, de forma que puedan utilizar las operaciones que este API expone.

1.8.4. Pruebas.

Se han seguido las siguientes estrategias y tácticas recomendadas para probar aplicaciones web (Pressman R. , 2010i) que recogen los principios básicos de pruebas y los recomendados para aplicaciones orientadas a objetos.

1.8.4.1. Prueba del modelo de contenido.

Se ha revisado el contenido de la web con el objetivo de descubrir errores tipográficos o gramaticales, errores en la precisión o completitud de la información y se ha comprobado la correcta estructura de la información presentada al usuario.

La base de datos no dispone de información a priori obtenida de fuentes de datos externas, salvo los datos de prueba que se ofrecen en un script sql.

Se han realizado pruebas en el ordenador local para asegurar la comunicación entre al base de datos y el interfaz de usuario final. Se han comprobado que los contenidos dinámicos de la base de datos se transmiten entre el usuario final y el servidor de la aplicación.

1.8.4.2. Prueba del intefaz del usuario.

El interfaz del cliente se ha probado también de forma manual, cada una de las funcionalidades que sus páginas web ofrecen.

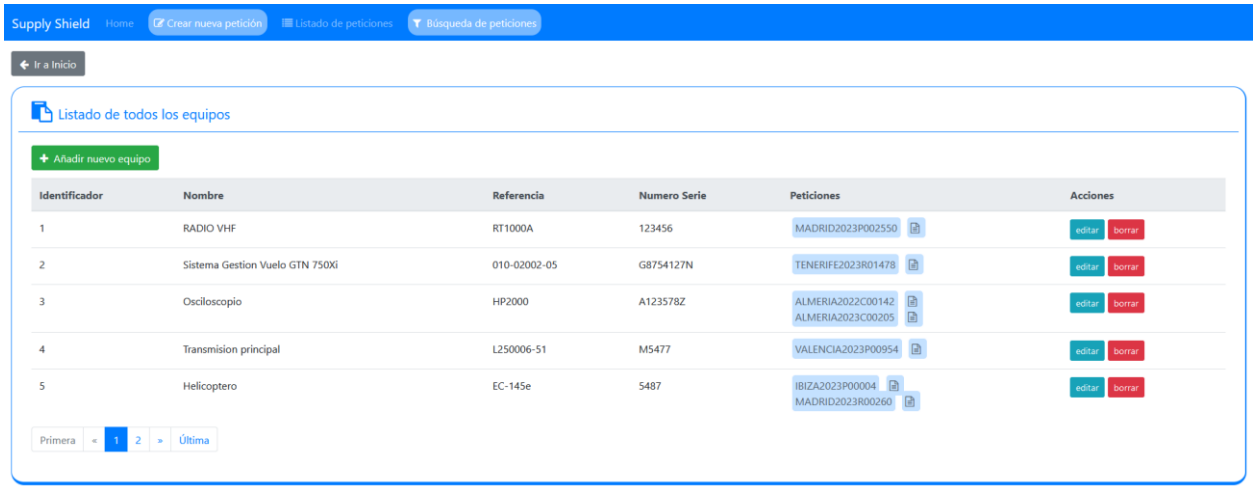

Figura 96: Probando los enlaces de la página listEquipo.html (fuente elaboración propia).

Se ha probado que es posible realizar todos los casos de uso con la interfaz disponible. Los vínculos funcionan correctamente y las etiquetas de los formularios identifican correctamente la información. Las ventanas pop-up de confirmación se muestran de forma adecuada.

1.8.4.3. Prueba de navegación.

Esta prueba comienza con la prueba del interfaz del cliente. Se comprueba que se posible realizar la navegación entre páginas propuesta en el mapa de navegación, los vínculos de página, los marcos que contienen información y las redirecciones de los menús o botones.

1.8.4.4. Pruebas de configuración.

Limitado por el limitado número de plataformas hardware disponibles, únicamente se ha comprobado el despliegue en la nube de la blockchain con un proveedor y la ejecución de la interfaz del cliente en los navegadores Firefox y Chrome.

En el ordenador local se ha probado la ejecución del microservicio servidor y cliente en Windows 10 y la blockchain en Linux Ubunto 20.04 ejecutándose bajo WSL.

1.8.4.5. Pruebas de API's.

Para probar el funcionamiento correcto de las api's expuestas por el servidor backend o la red blockchain se ha utilizado la herramienta Postman.

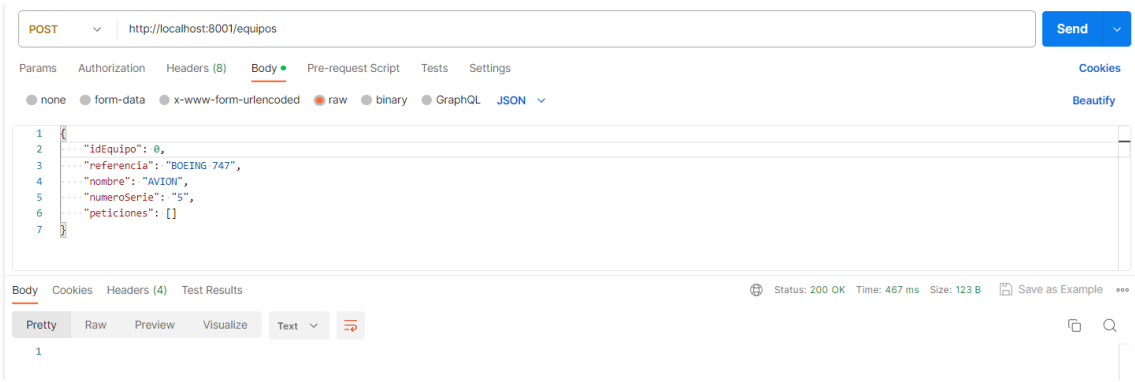

Figura 97: Vista de Postman realizando una petición POST al servidor (fuente elaboración propia).

Se han realizado pruebas de todos los endpoint del api del backend y de la red blockchain utilizando la herramienta, que, de forma muy sencilla, permite realizar peticiones get, put, post y delete, añadiendo información en el body de la petición cuando es necesario. Unicamente colocar la dirección que se quiere probar en la barra de direcciones y si es necesario información en el body. Con solo pulsar el botón "SEND" la aplicación nos da detalles de la respuesta obtenida. Se han probado completamente las api de forma manual.

1.9. Fase de Transición.

La fase de transición incluye las últimas etapas del proceso general de construcción y la primera del despliegue general, se realiza una entrega al usuario para que proporcione realimentación. El usuario dispone del software para que detecte defectos y proponga cambios. Se entrega un manual de utilización para el usuario, al final de esta fase el software ya es un producto que se lanza.

En este proyecto se ha concentrado el despliegue de la aplicación en esta fase. A continuación se entrega un pequeño manual de uso y se explica el proceso para realizar el despliegue de una red blockchain en un proveedor real, (Kaleido) con dos miembros y sus respectivos nodos ejecutando Hyperledger FireFly. Ya se explicó en el apartado de implementación de la red blockchain, cómo hacelo en el ordenador local para desarrollo. Seguidamente se explica como desplegar los servicios cliente servidor en el ordenador en local.

1.9.1. Manual de uso.

La interfaz del cliente dispone de un menu superior desde el que en cualquier página se puede el usuario redirigir a las otras, si no encuentra el camino.

1.9.1.1. Página principal.

Se puede observar en la figura 98, tanto el menú superior para el acceso a las paginas relacionadas con las peticiones de mantenimiento, como los botones de acceso a las mismas funcionalidades.

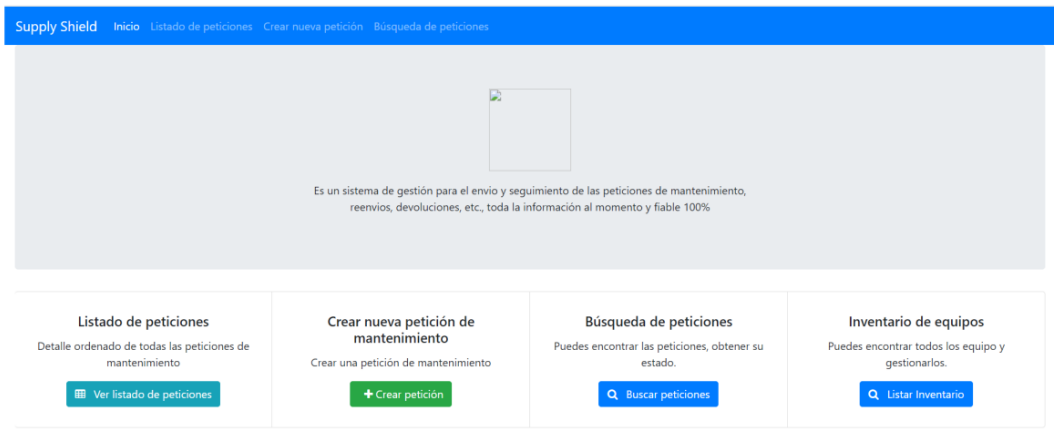

Figura 98: Vista de la página principal (fuente elaboración propia).

Pulsando en el botón de abajo a la derecha se tiene acceso al inventario de los equipos disponibles.

1.9.1.2. Listado de peticiones.

Pulsando en el botón "Listado de peticiones" se muestra una pagina con el listado de las peticiones. Desde este listado es posible eliminar una petición si se pulsa el botón borra y se confirma la operación, se puede editar la petición de mantenimiento para actualizar datos o añadirlos en función del avance en la situación de la petición de mantenimiento, o también consultar la información disponible si se pulsa en el botón con el índice de la petición de mantenimiento.

Si se desea salir de esta pantalla se puede conseguir bien a través del botón “Ir al Inicio”, bien a través de los enlaces del menú superior

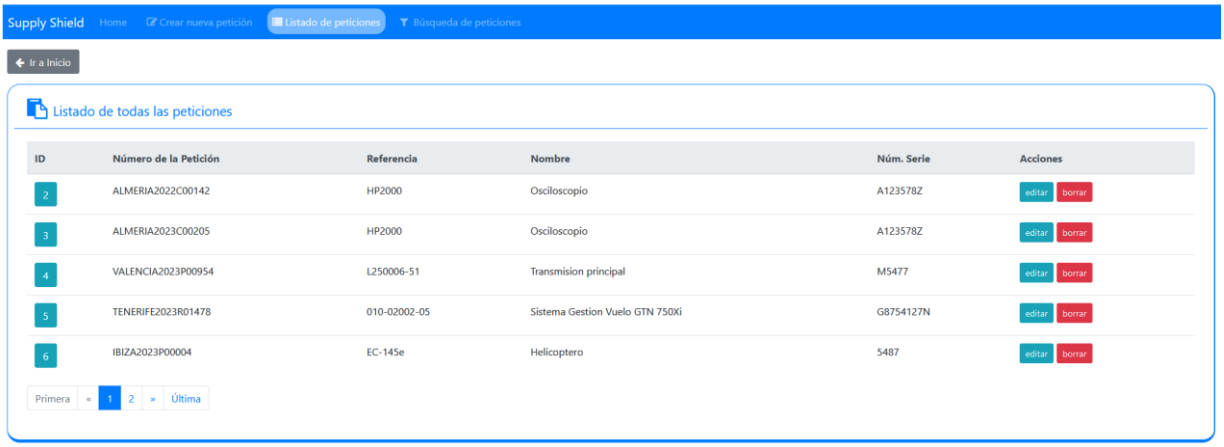

Figura 99: Vista de la página mostrando peticiones de mantenimiento. (fuente elaboración propia).

1.9.1.3. Consulta de una petición.

Pulsando en el botón azul con el índice de la petición, es posible visualizar la información existente.

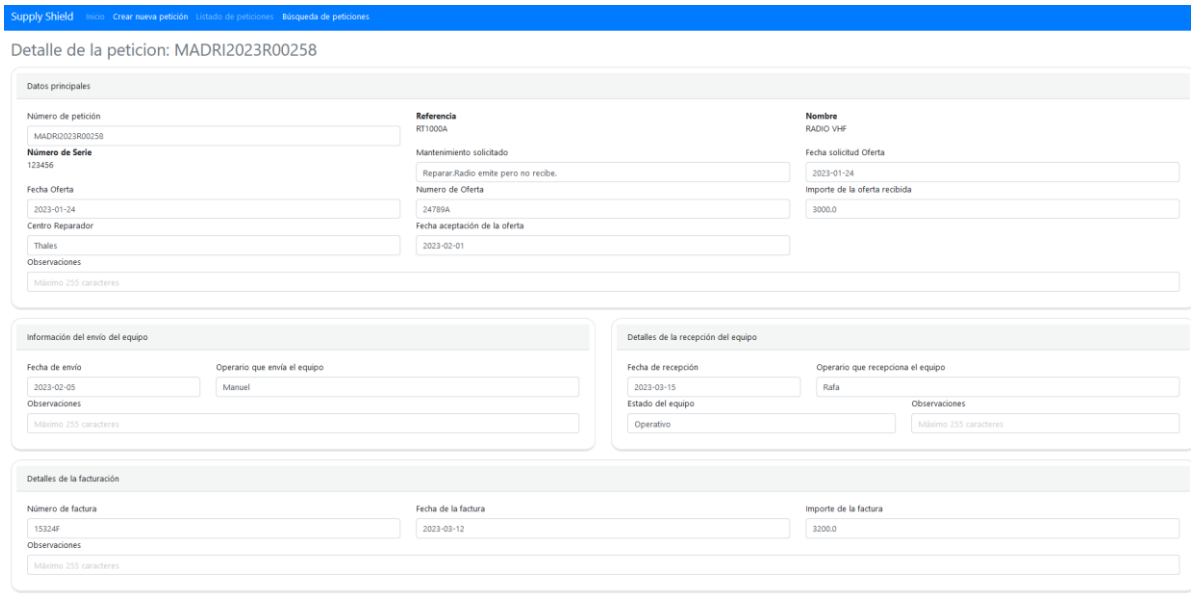

Figura 100: Vista de la página mostrando la información de una petición de mantenimiento. (fuente elaboración propia).

1.9.1.4. Actualización de una petición.

Para actualizar la información de una petición pulsar "Editar" en el botón correspondiente a la petición mostrada en el listado de peticiones.

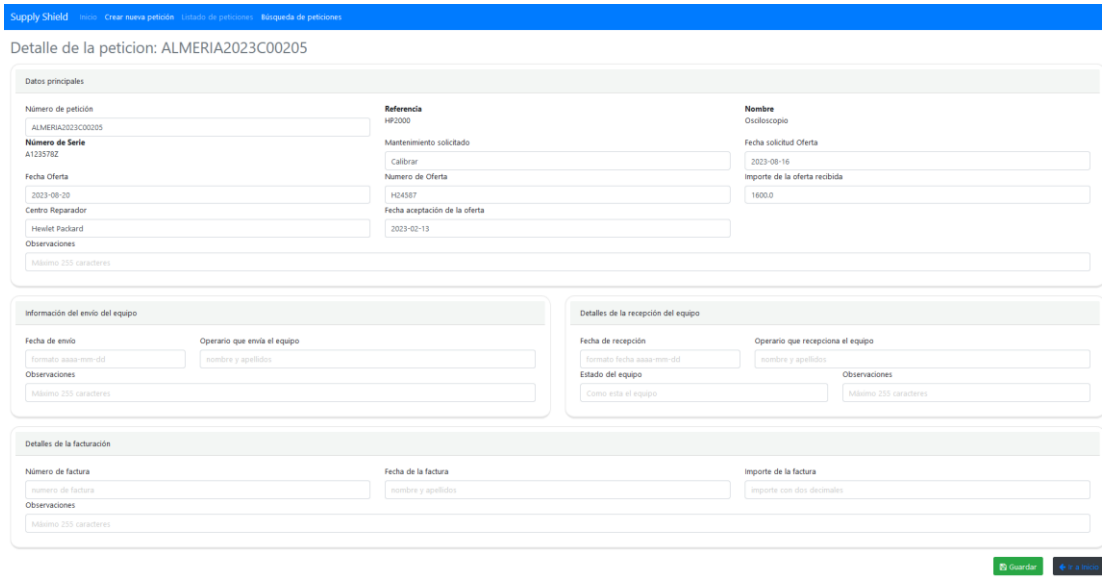

Figura 100: Vista de la página permite actualizar la información de una petición de mantenimiento. (fuente elaboración propia).

Si en este caso el equipo lo recoge un transportista para llevarlo al centro reparador, se registra la fecha de envío, el operario que lo entrega y si es preciso se rellenan observaciones. Para almacenarlo se pulsa en el botón de color verde "guardar".

De la misma forma, una vez regrese el equipo del mantenimiento se registra la información en el marco detalles de la recepción del equipo, o bien en el momento de realizar la facturacion, se incorpora la información en el marco detalles de la facturación.

1.9.1.5. Creación de una petición.

Si se pulsa el botón "crear petición" en la página principal se muestra el formulario de la figura anterior.

Una vez rellenados los detalles de la petición mínimos para realizar la petición (número, motivo y centro reparador de la petición, se puede pulsar el botón guardar para almacenar la petición de mantenimiento.

Posteriormente, se puede añadir el equipo que va a ser objeto de la petición de mantenimiento.

1.9.1.6. Buscar una petición de mantenimiento.

Pulsando en el enlace del menú superior o en el botón de la página principal "buscar peticiones" es posible buscar peticiones de mantenimiento por número de petición, número de serie o bien referencia del equipo rellenando la información buscada y pulsando el botón azul "buscar por" correspondiente.

Se mostrará la página del listado de peticiones con todas aquellas que cumplen el criterio de búsqueda fijado. A partir del listado ya se pueden realizar operaciones de actualización, consulta o borrado.

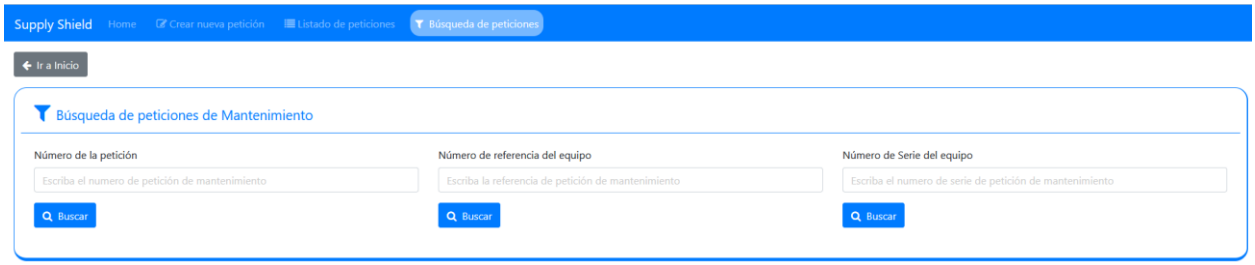

Figura 101: Vista de la página para buscar peticiones de mantenimiento (fuente elaboración propia).

1.9.1.7. Inventario de equipos.

Si se pulsa el botón "Listar Inventario" se puede ver un listado con todos los equipos y las peticiones de mantenimiento que se le han realizado.

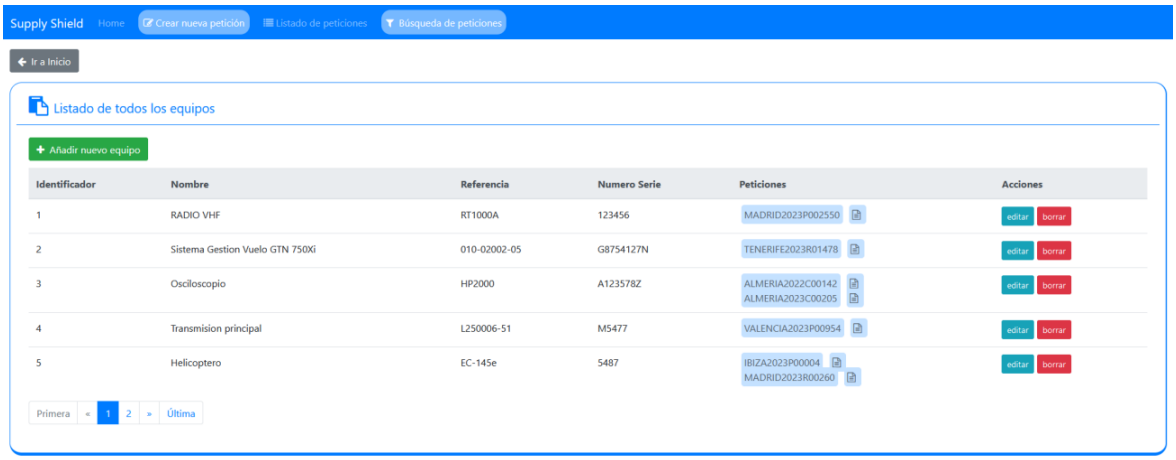

Figura 102: Vista del listado de equipos (fuente elaboración propia).

Desde este listado es posible editar la información del equipo o bien borrar el equipo una vez confirmada la operación.

De igual forma es posible añadir un nuevo equipo pulsando en el botón azul "Añadir un nuevo equipo".

Figura 103: Vista del formulario para actualizar o añadir equipos (fuente elaboración propia).

1.9.2. Despliegue de la red Blockchain.

En el punto 1.8.3.5 se ha explicado cómo se realiza la instalación del entorno de FireFly para realizar el desarrollo y pruebas de una red blockchain en local. En este punto se explica cómo realizar una red de dos miembros y el despliegue del smart contract en un proveedor en internet, Kaleido (Kaleido, 2022d). Se ha utilizado una cuenta gratuita que está limitada a dos nodos, suficiente para este proyecto. Es muy similar, aunque en este caso, se ha introducido la variación de realizar el compilado del smart contract en la propia plataforma del proveedor. La razón es evitar errores de compatibilidad entre la versión del compilador y la version de FireFly utilizada, como ocurrió durante el despliegue en local.

1. Crear la red.

Una vez se accede a la plataforma del proveedor con usuario y password, se muestra un tablero de control. Para comenzar el proceso de creación de la red seleccionar el botón Create Network. El sistema solicita un nombre, Student en este caso, y seleccionar una region para el almacenamiento de la red. La red la crea con un miembro pero para este proyecto se crean dos más, OrganizacionCliente y CentroReparador. Para ello seleccionar el botón "ADD MEMBERSHIP" y añadir el nombre deseado.

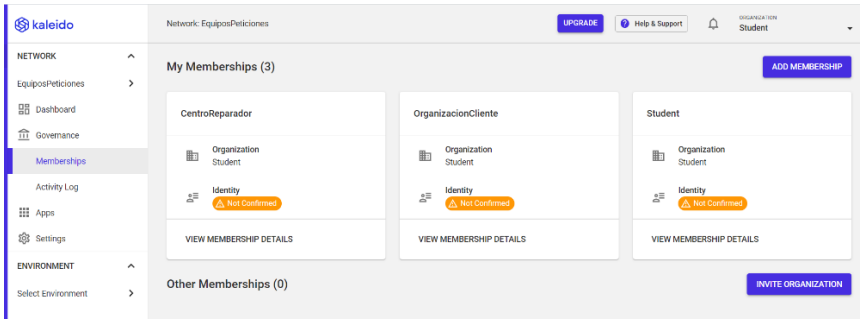

Figura 105: Miembros de la red CentroReparador y OrganizacionCliente (fuente elaboración propia).

2. Configuración del entorno.

Para iniciar la configuración del entorno necesario seleccionar en el menu izquierdo ENVIRONMENT/Select Environment/Add environment.

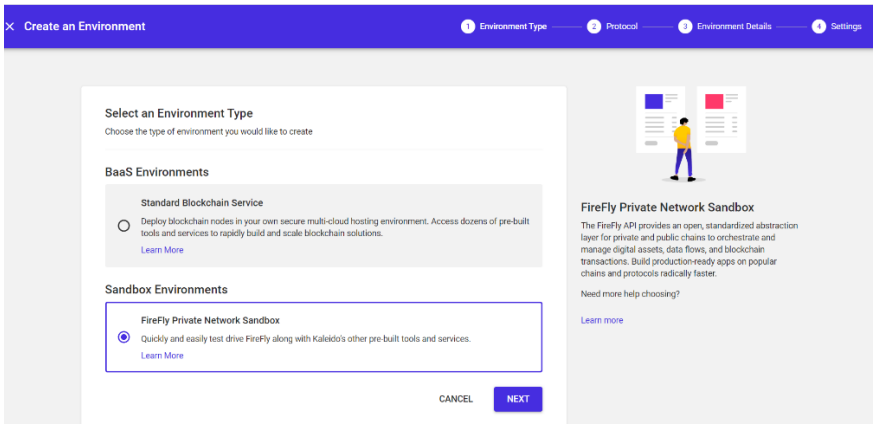

Figura 106: Inicio creación entorno (fuente elaboración propia).

Seleccionar FireFly Private Network Sanbox y el botón NEXT.

En el formulario que muestra seleccionar el tipo de red, ethereum en este caso, añadir un nombre (data-transfer se ha elegido) y desplegar en la nube. Como proveedor seleccionar Geth y el algoritmo de consenso POA (Probe of Authority).

Al seleccinar el botón FINISH ya está configurado el entorno.

3. Crear los nodos FireFly de la red.

El siguiente paso es el nodo de FireFly de cada miembro de la red.

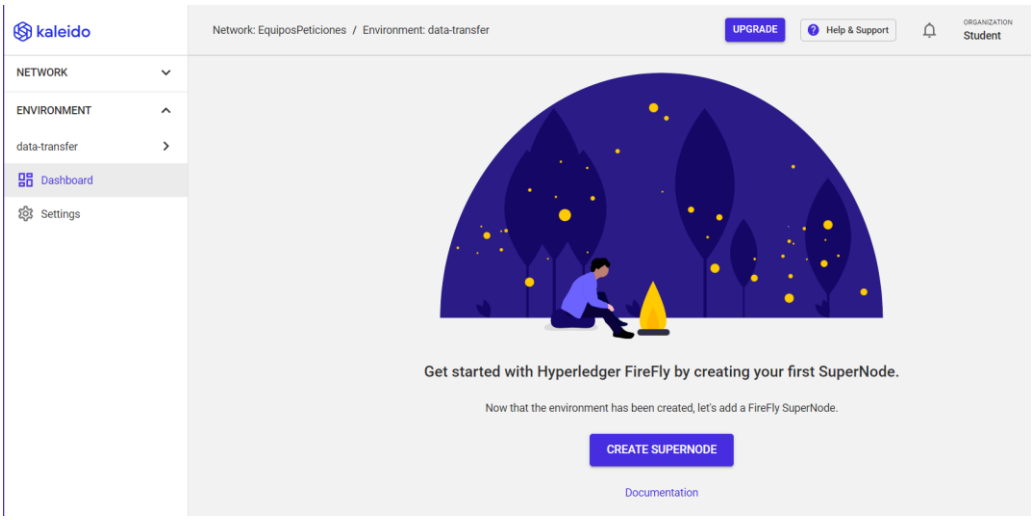

Figura 107: Inicio creación nodos FireFly (fuente elaboración propia).

Una vez seleccionado el botón "CREATE SUPERNODE", se selecciona el miembro de la red al que le vamos a añadir el nodo, elegir un nombre para el nodo y seleccionar el botón "NEXT".

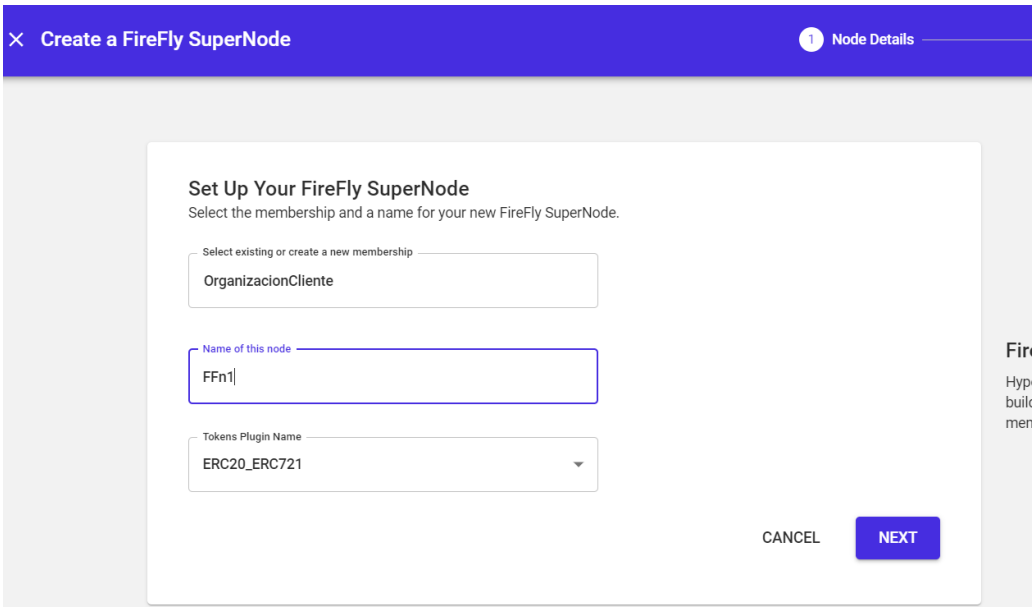

Figura 108: Selección miembro al que se va a crear el nodo (fuente elaboración propia).

El siguiente formulario define el almacenamiento IPFS asociado al nodo, como se ve en la figura XX.

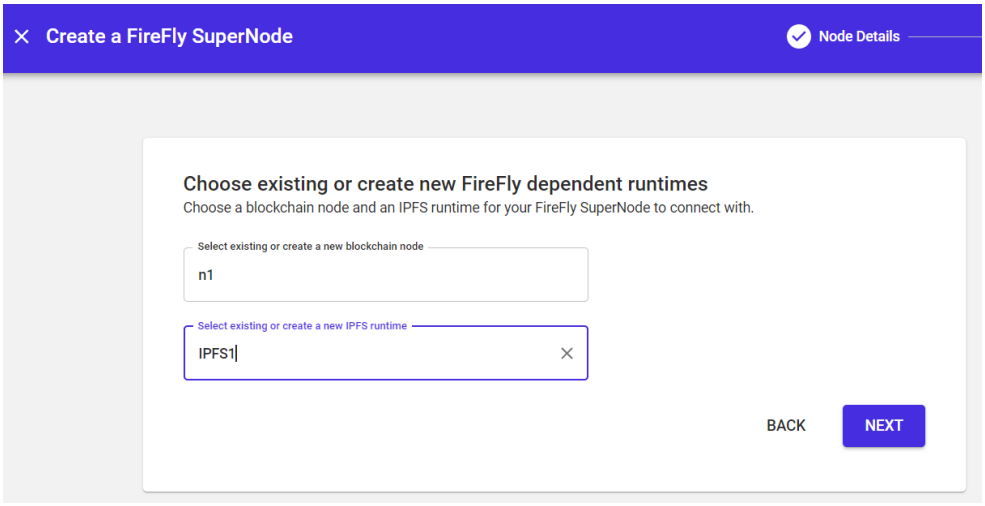

Figura 109: Selección almacenamiento IPFS (fuente elaboración propia).

Una vez seleccionado de nuevo el botón "NEXT", ya solo queda seleccionar la potencia de computación y pulsar el botón "FINISH" para que el nodo FireFly del miembro OrganizacionCliente sea creado por el sistema.

El proceso se repite para la creación del nodo FireFly del miembro de la red CentroReparador. Se puede observar en la figura XX como se muestra en el dashboard de la plataforma de Kaleido una vez creados los dos nodos.

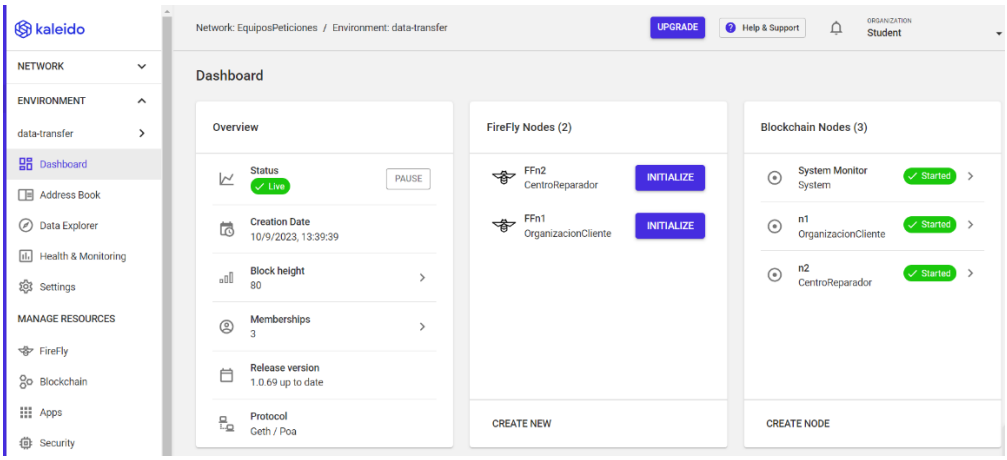

Figura 110: Visualización de los nodos FireFly (fuente elaboración propia).

Sólo queda pulsar el botón "INITIALIZE" de cada nodo para que se pongan e funcionamiento.

4. Despliegue del smartcontract.

Si seleccionamos uno de los nodos, FFn1 por ejemplo, se tiene acceso a información de ese nodo. Los endpoint del api, la interfaz del sandbox, el document swagger, etc. Seleccionando FireFly Sandbox/Open se abre una nueva pestaña en el navegador con el sandbox, idéntico al de la figura 84 89 ó 90, que se desplegó en local.

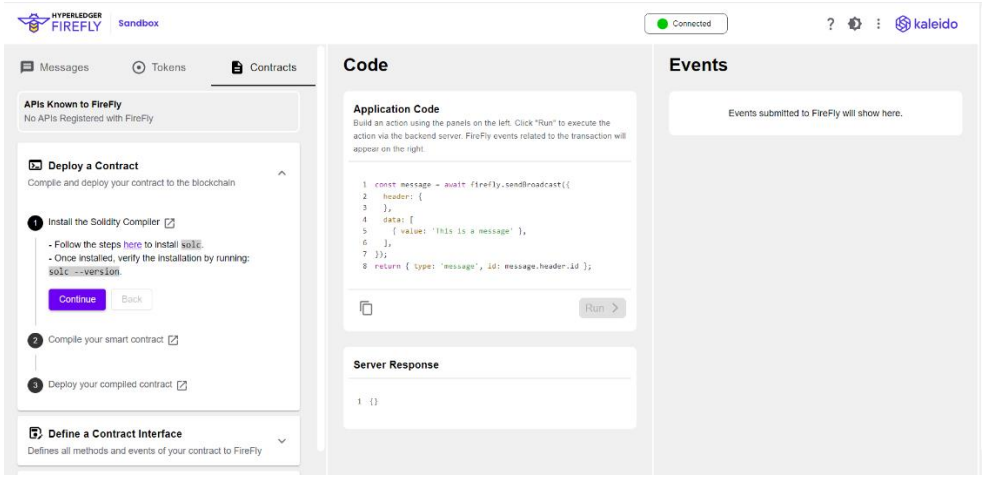

Figura 111: Sandbox del nodo FireFly FFn1 (fuente elaboración propia).

El proceso para el despliegue del smartcontract y la generación y registro de su api, es muy parecido al descrito en el punto 1.8.3.5.

a) Añadir el código fuente del smartcontract y desplegarlo.

Seleccionando en el menu izquierdo "NETWORK/Apps" se visualiza en el dashboard las aplicaciones que están disponibles. Seleccionando el botón "ADD NEW APP" para añadir una nueva.

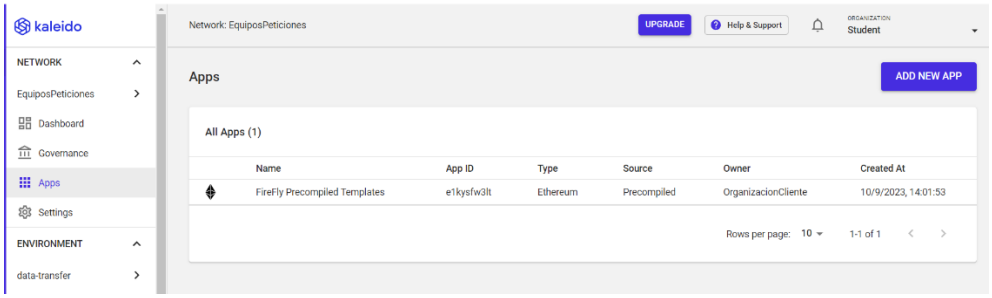

Figura 112: Vista aplicaciones disponibles (fuente elaboración propia).

En el formulario que se muestra, se selecciona ethereum como tipo de red donde va a desplegarse el smartcontract y el miembro a través del cual se va a realizar la operación. En vez de entregar el código compilado, como se hizo en el despliegue en local, se elege la opción para entregar el código fuente.

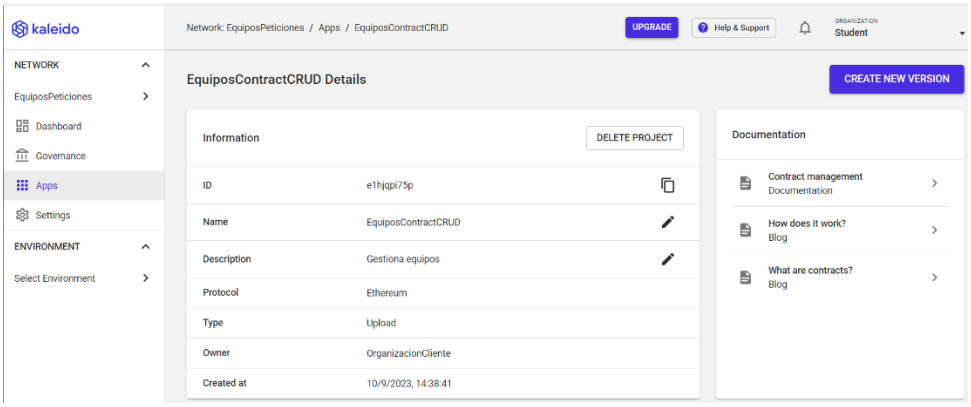

Figura 113: Vista de la configuración del smartcontract a desplegar (fuente elaboración propia).

Para entregar el código fuente se selecciona el botón "CREATE NEW VERSION" de la figura anterior, seleccionar el miembros de la red que lo va a desplegar y una descripción. A continuación el sistema muestre el formulario que permite seleccionar el fichero con el Código fuente situado en el ordenador local, equiposContractCRUD.sol en este caso. El sistema detecta el nombre del smartcontract definido en el fichero para que se seleccione. Se puede elegir la version del compilador y la máquina virtual ethereum pero la intención es dejar que el sistema lo detecte automáticamente a partir del código fuente asi que se sugiere no seleccionar ninguno de los dos. Se pulsa en el botón "FINISH" y el sistema compilará el contrato automáticamente.

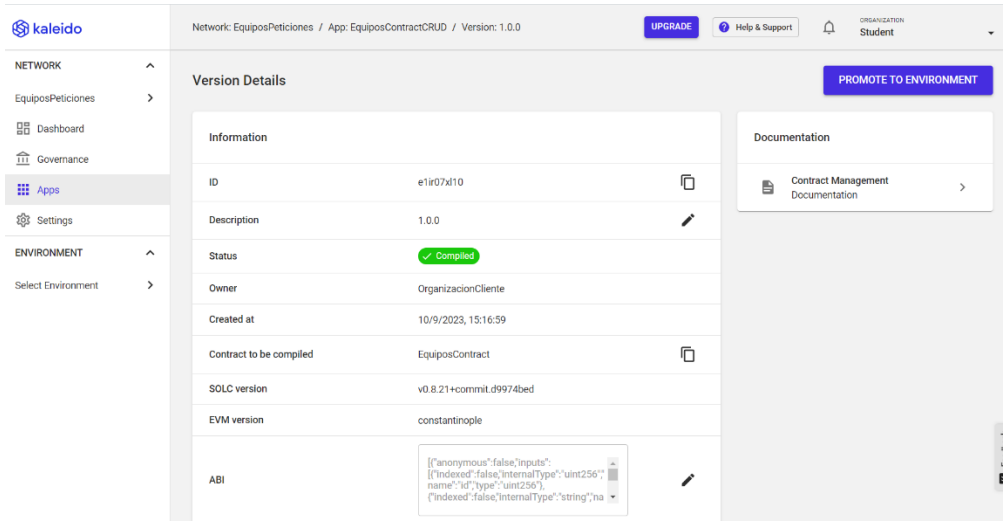

Figura 114: Vista de la smart contract una vez compilado (fuente elaboración propia).

Se observa en la figura anterior los detalles de la compilació como por ejemplo el contenido ABI que después hace falta durante el despliegue del contrato a través del sandbox del miembros de la red OrganizacionCliente. Ahora se selecciona el botón "PROMOTE ENVIRONMENT" y se selecciona el entorno donde se va a desplegar, en este caso data-transfer, que es nombre que le hemos antes se dió al entorno. También se añade un nombre corto para definer el endpoint del smart contract.

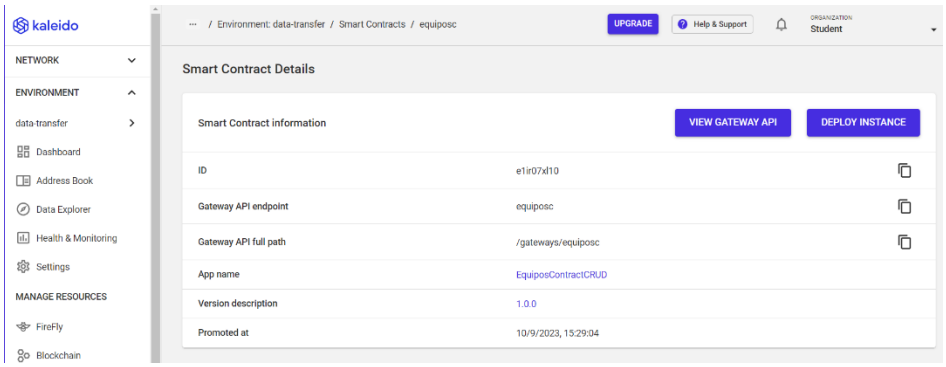

Figura XX: Vista de la smart contract una vez compilado (fuente elaboración propia).

Si se selecciona el botón "VIEW GATEWAY API" y en el formulario que se muestre el botón "VIEW API", se creará una instancia en el nodo 1 del miembre OrganizacionCliente y se mostrará la definición de la api en un documento swagger que además permite interactuar con la red.

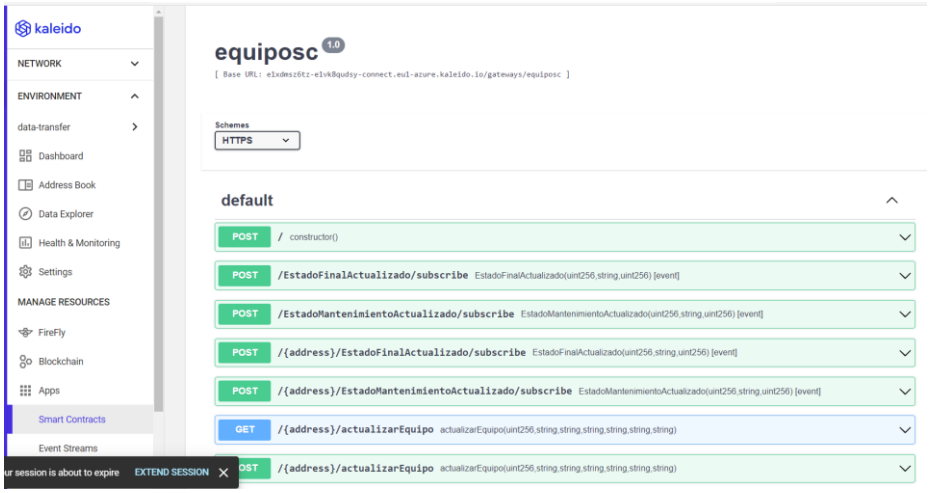

Figura 115: Vista de la definición del API del smart contract desplegado (fuente elaboración propia).

Ejecutando una consulta en el constructor y registrando un nombre sencillo para la ruta del endpoint, por ejemplo eq123:

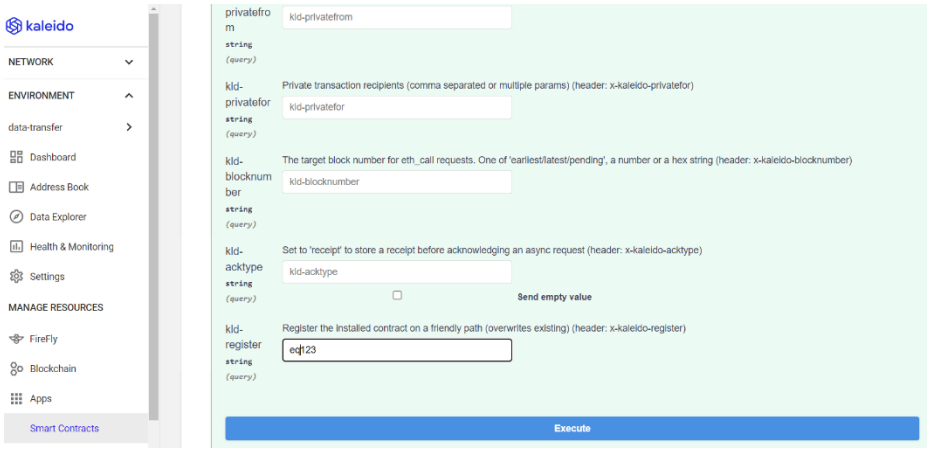

Figura 116: Ejecutando una consulta desde el constructor (fuente elaboración propia).

El servidor responde positivamente con los detalles de la respuesta (código 200)

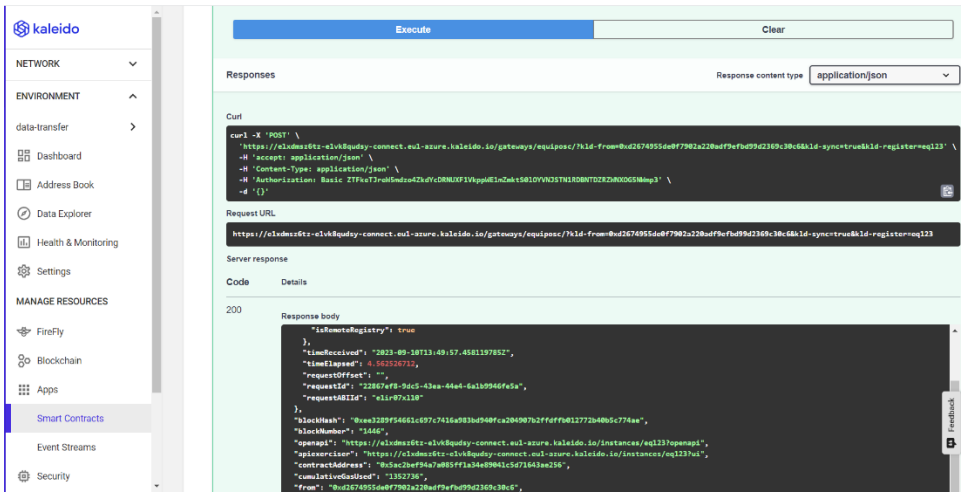

Figura 117: Respuesta del servidor (fuente elaboración propia).

En los detalles del smart contract se puede ver la ruta del endpoint o la dirección del smart contract una vez registrado en la red blockchain.

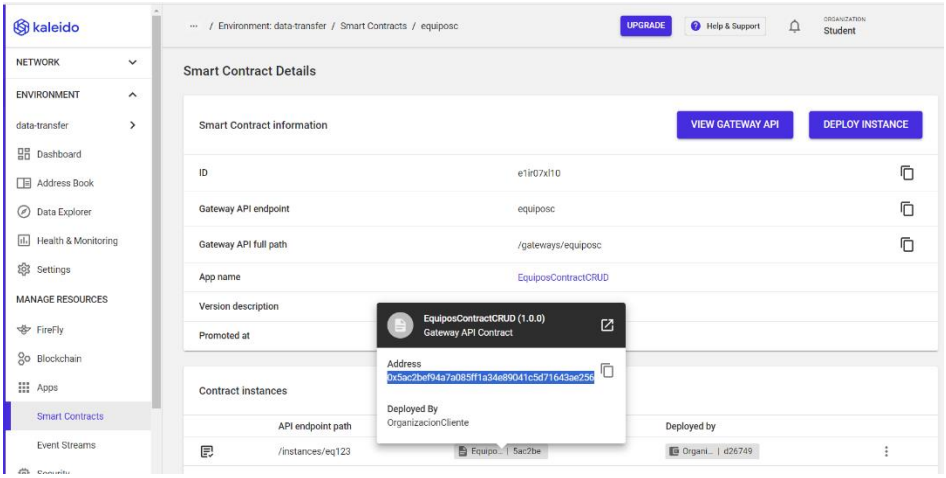

Figura 118: Detalles del smart contract (fuente elaboración propia).

A partir de aquí el proceso de definición de la interfaz del api, su registro y la definición de los listeners de los eventos es idéntica que los definido en el punto 1.8.3.5 y se realza a través del sandbox del miembro OrganizacionCliente, por ejemplo.

b) Definición del interface del contrato.
A través del sandbox se selecciona el formato ABI, se añade un nombre y una version. Por ultimo se sustitue el ABI por defecto por el del contrato (figura XX). Seleccionando el botón "RUN" se ejecuta el métodos generateContractInterface.

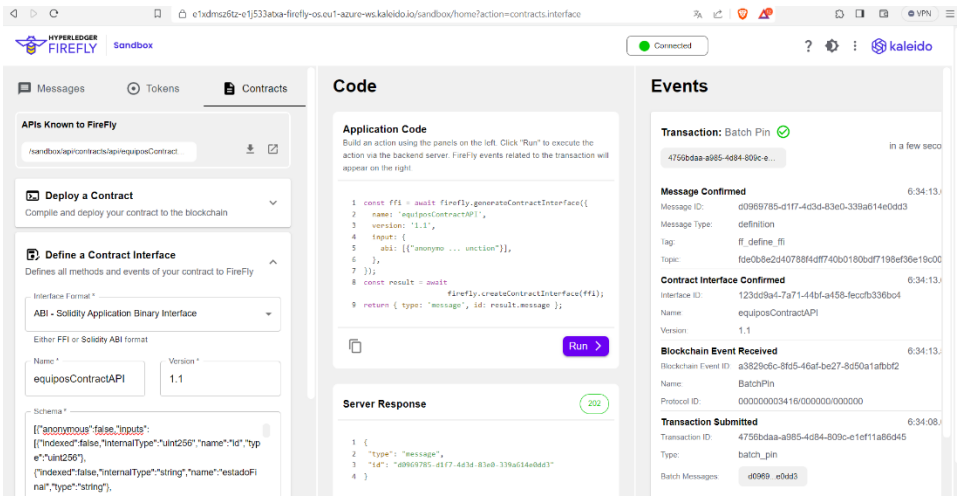

Figura 119: Vista del dashboard con la interface generada (fuente elaboración propia).

c) Registro del API con la dirección del contrato.
Seleccioando el siguientes paso "Register a Contract API" se despliega un formulario que permite ingresar un nombre para el API, el address del contrato (figura anterior) y seleccionar la interfaz definida en el paso b). Por ultimo se vuelve a seleccionar "RUN".

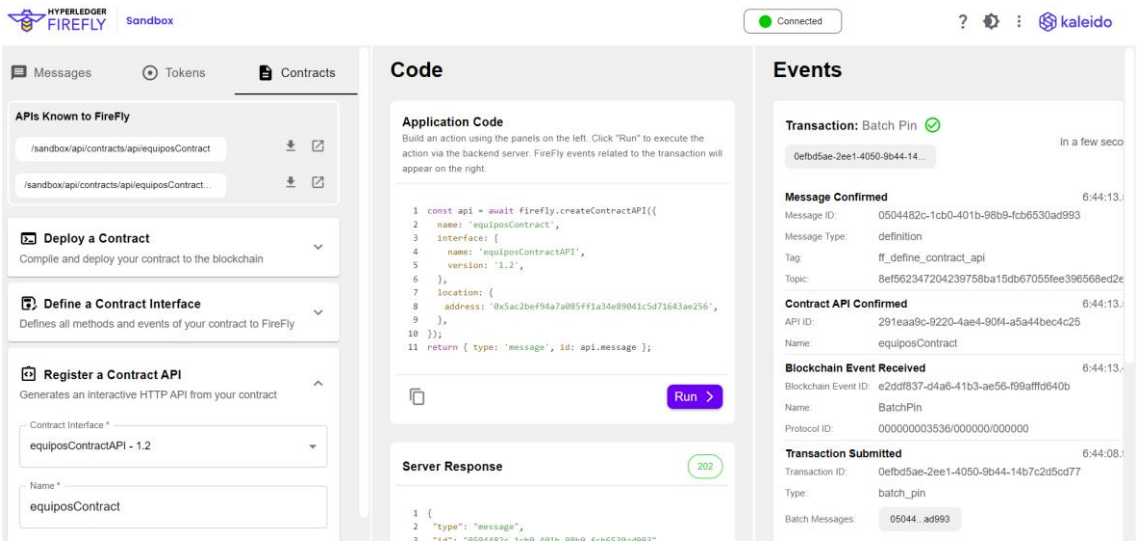

Figura 120: Vista del dashboard con la interface registrada (fuente elaboración propia).

Se puede observar como se activa el menu superior derecho "APIs Known to FireFly" en la figura anterior. Se ven dos API's porque se generaron dos versions, la 1.1 y la 1.2. Si se pulsa en botón de la primera (la 1.2) se despliega el swagger del api.

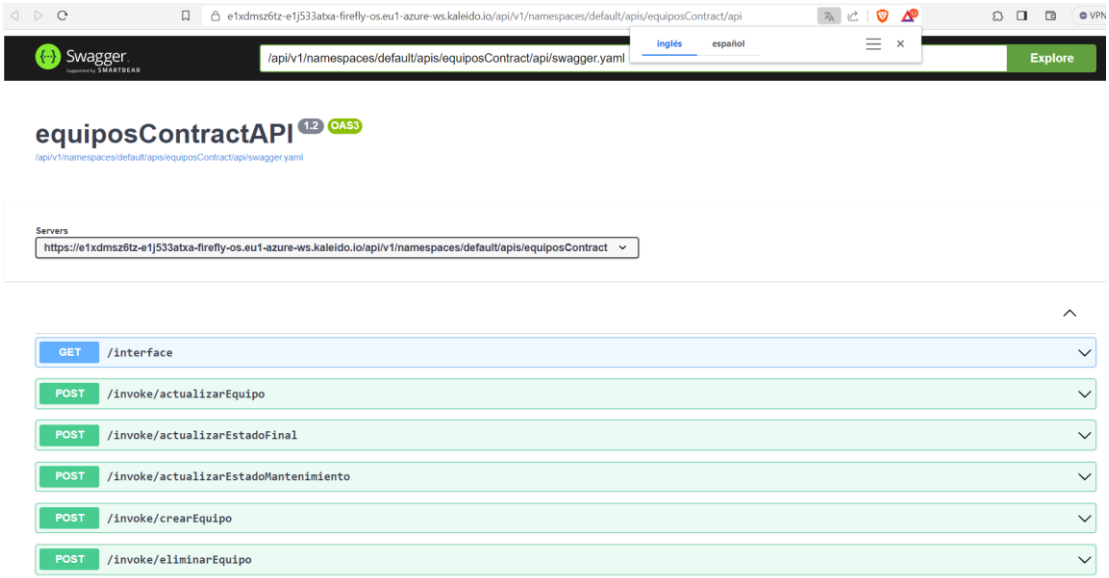

Figura 121: Vista de la definición complete del api generada, definida y registrada (fuente elaboración propia).

d) Registrar los eventos del contrato (listeners).

Por ultimo queda seleccionar "Register a Contract Listener" que muestre un formulario para seleccionar los eventos del contrato para registrarlos, dándoles un nombre y pulsando en el botón "RUN".

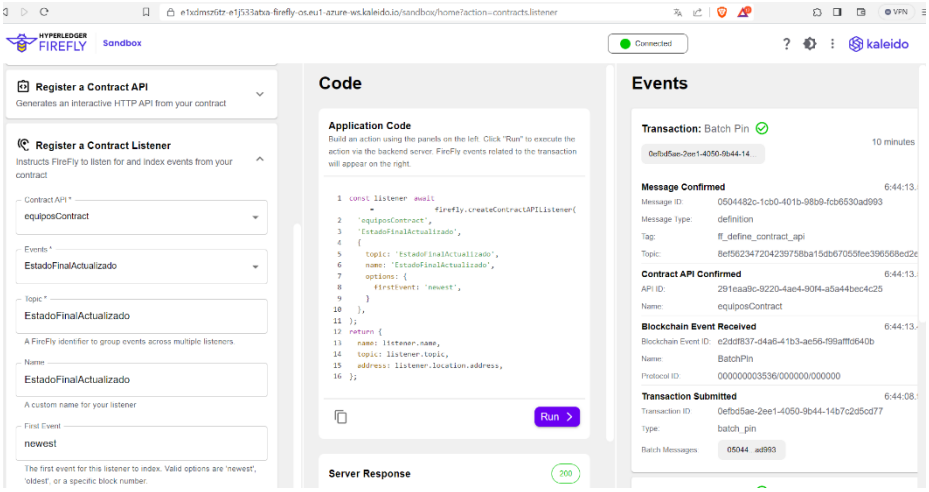

Figura 122: Vistas del dashboard con un listener registrado (fuente elaboración propia).

Si se accede al sandbox del miembro CentroReparador se observa que el API está igualmente disponible en el menú "APIs Known to FireFly".

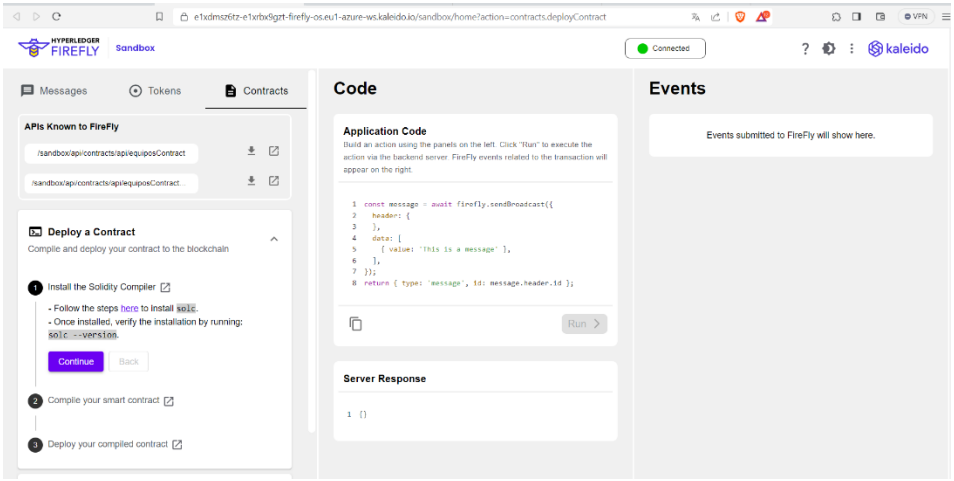

Figura 123: Vista del sandbox del miembro CentroReparador (fuente elaboración propia).

1.9.3. Despliegue del microservicio backend.

El despliegue del backend se realiza creando un contenedor docker para ejecutar la base de datos MySql, y conectándolo en red con un segundo contenedor que ejecuta el apirest.

1.9.3.1. Contenedor base de datos MySQL.

1.- Descarga imagen MySQL.

En primer lugar, es necesario descargar la imagen de mysql para lo que se utiliza el comando docker pull mysql.

2.- Creación del contenedor.

Se crea un contenedor utilizando la imagen mysql, que se ejecuta en segundo plano (-p) de nombre mysql_container con la password de usuario root indicada en el comando. También se fija el juego de caracteres.

docker run -d -p 13306:3306 --name mysql_container -e MYSQL_ROOT_PASSWORD=secret mysql --character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci .

3.- Creación de la base datos.

Para crear la base de datos es necesario acceder al contenedor abriendo una consola interactive con el commando docker exec -it mysql_container mysql -uroot -p .

Se abre un prompt mysql, una consola para interactuar con la base datos utilizando los comandos reconocidos por la base de datos. Para crearla se copia el contenido del script .sql y se pega en la consola. Las órdenes SQL se ejecutan automáticamente.

Para completar la base de datos con algunos datos se copia el contenido del script .sql y se pega en la consola de la misma forma que durante la creación.

4.- Comprobación del funcionamiento del contenedor.

Para comprobar el buen funcionamiento del contenedor, se modifica el fichero application properties del proyecto y se sustituye la variable datasource.url para que apunte al puerto 13306 donde está escuchando el contenedor.

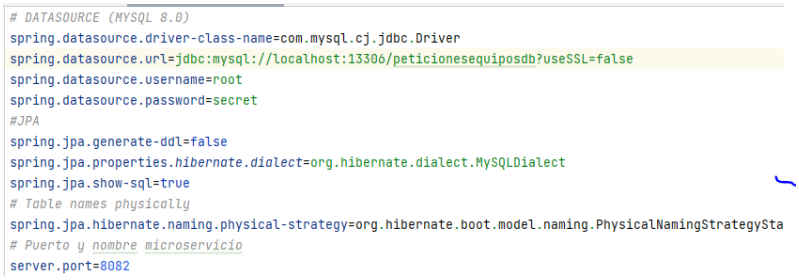

Figura 124: Vista del fichero de configuración apuntando al contenedor (fuente elaboración propia).

Ahora en el entorno IntelliJ, se puede configurar una nueva fuente de datos mysql que apunte al puerto del contenedor localhost:13306, y ejecutar una consulta a la tabla peticiones.

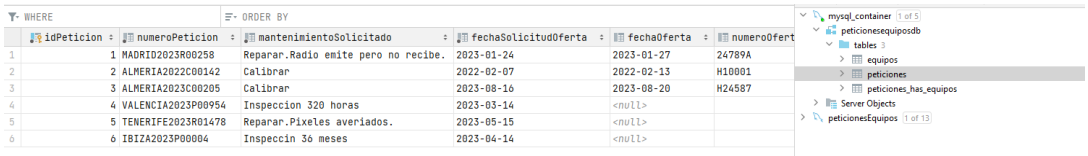

Figura 125: Vista de la tabla peticiones en IntelliJ (fuente elaboración propia).

Por ultimo, se comprueba el funcionamiento del api rest con la herramienta Postman, realizando peticiones al api en el puerto 8082, que es el puerto definido en el fichero de configuración del microservicio.

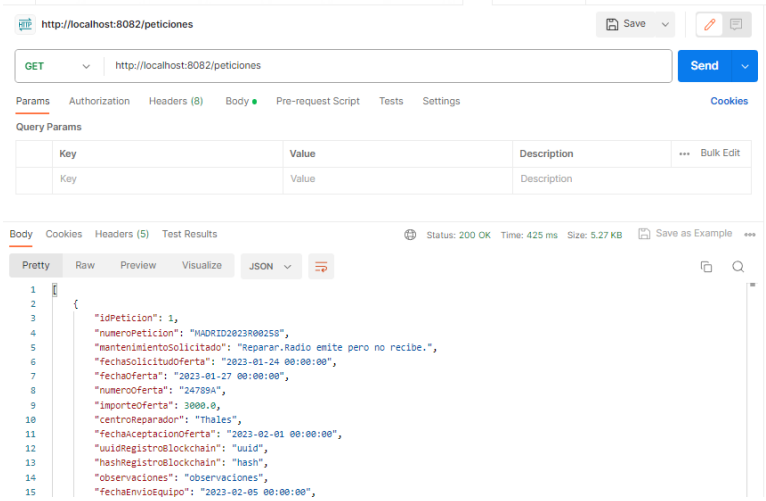

Figura 126: Vista en postman de la consulta de las peticiones de la base de datos (fuente elaboración propia).

Ya solo queda conectarlo a una red.

5.- Creación y conexión a la red

Se puede comprobar que el contendor mysql_container está conectado a la red por defecto bridge ejecutando el comando docker inspect bridge.

Primero es necesario desconectarlo de la red bridge para lo que se ejecuta el comando docker network disconnect bridge mysql_container.

Se crea la nueva red my-net con el commando docker network create --driver bridge my-net .

Por ultimo se conecta el contenedor mysql_container a la red con el comando docker network connect my-net mysql_container.

Se puede comprobar la correcta conexión a la red ejecutando del comando docker inspect my-net.

1.9.3.2. Contenedor del microservicio del backend.

Ahora es necesario crear un contenedor docker que ejecute el archivo .jar con la aplicación que expone el api rest del backend.

1.- Modificación fichero application properties.

Es necesario modificar la variable datasource.url para que apunte al contenedor mysql_container en vez de al localhost.

```
# DATASOURCE (MYSQL 8.0)
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://mysql_container/peticionesequiposdb?useSSL=false
spring.datasource.username=root
spring.datasource.password=secret
#JPA
spring.jpa.generate-ddl=false
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
spring.jpa.show-sql=true
# Table names physically
spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl
# Puerto y nombre microservicio
server.port=8082
spring.application.name=servicio-peticiones-equipo
```

Figura 127: Vista de application properties definiendo la variable datasource.url (fuente elaboración propia).

2.-Creación del fichero JAR del proyecto.

Para la creación del fichero JAR del proyecto con IntelliJ es necesario ir al menú File/Project Structure/Proyect Settings/Artifacts , pulsar en el símbolo + y seleccionar la opción "From modules with dependencys".

Se mostrará un formulario para seleccionar la clase principal y pulsar en OK.

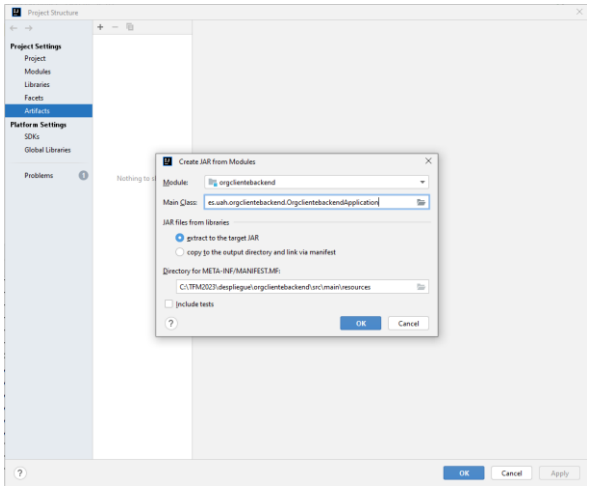

Figura 128: Selección de la clase principal (fuente elaboración propia).

El siguientes paso es compilarlo para lo que es necesario seleccionar el menu Build/Build Artifacts.

Se mostrará un menu Build/Action en el que es necesario elegir Build:

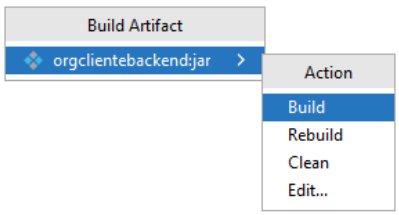

Figura 129: Ejecutando la compilación (fuente elaboración propia).

Una vez compilado el archivo JAR del proyecto se mostrará en la ruta out/artifacts:

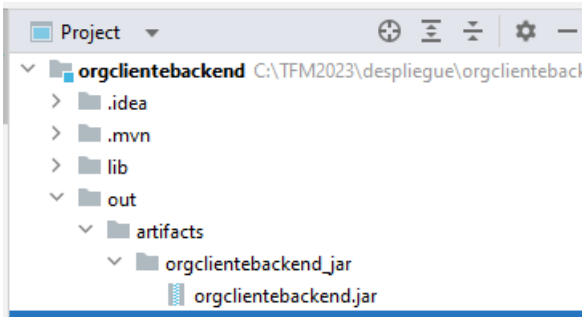

Figura 130: Ruta con el fichero jar generado (fuente elaboración propia).

2.- Creación de una imagen.

Para crear una imagen con el proyecto se copia el archivo jar a la ruta despliegue/orgclientefrontend/target y se añade el fichero Dockerfile con el Código de la siguientes figura:

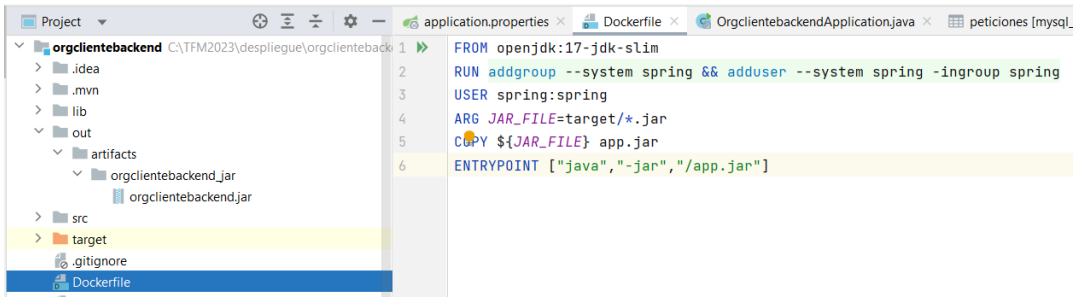

Figura 131: Ruta con los fichero Dockerfile y jar generados (fuente elaboración propia).

Por ultimo desde el directorio del proyect despliegue/orgclientebackend se ejecuta el comando que crea la imagen "docker build -t backend/backend_service:V1 . " .

Se puede utilizar el comando docker images para ver la imagen creada.

Figura 132: Vista de la imagen creada (fuente elaboración propia).

3.-Levantar contenedor con api rest y conectarlo a la red my-net.

Ya solo queda levantar un contenedor con la imagen creada, y conectarlo a la red my-net con el comando docker run -d --network my-net -p 18082:8082 --name apirest_container cf74621eea96 (docker run --network my-net -d -p 18082:8082 --name name_container ImageID).

Figura 133: Vista de los dos contenedores en red (fuente elaboración propia).

1.9.4. Despliegue del microservicio frontend.

1.- Modificación de clases ServiceImpl de la capa service.

Estos ficheros (EquiposServiceImpl y PeticionesServiceImpl) apuntan las rutas del backend a localhost, pero ahora se está ejecutando en un contenedor de nombre apirest_container dentro de la red my-net, hay que redirigir la ruta de acceso al backend a apirest_container.

```java
@Service
public class EquiposServiceImpl implements IEquiposService{
    /** Implementa toda la lógica de negocio de los métodos de  la interfaz IEquiposService
     haciendo uso de los métodos de la interfaz RestTemplate que se ha importado.**/

    @Autowired
    RestTemplate template;

    8 usages
    String url = "http://apirest_container/equipos";/*Direccion base de microservicio backend al que va a realizar peticiones*/
```

Figura 134: Modificación de la clase EquiposServiceImpl (fuente elaboración propia).

2.-Creación del fichero JAR del proyecto.

Para la creación del fichero JAR del proyecto con IntelliJ es necesario ir al menú File/Project Structure/Proyect Settings/Artifacts , pulsar en el símbolo + y seleccionar la opción "From modules with dependencys".

Se mostrará un formulario para seleccionar la clase principal y pulsar en OK.

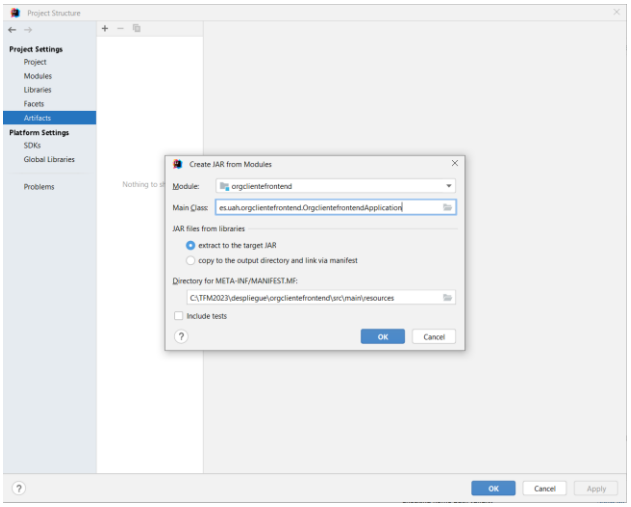

Figura 135: Selección de la clase principal (fuente elaboración propia).

El siguientes paso es compilarlo para lo que es necesario seleccionar el menu Build/Build Artifacts.

Se mostrará un menu Build/Action en el que es necesario elegir Build:

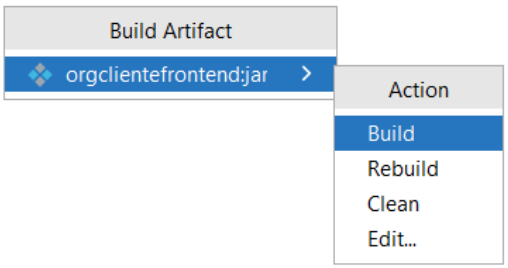

Figura 136: Ejecutando la compilación (fuente elaboración propia).

Una vez compilado el archivo JAR del proyecto se mostrará en la ruta out/artifacts:

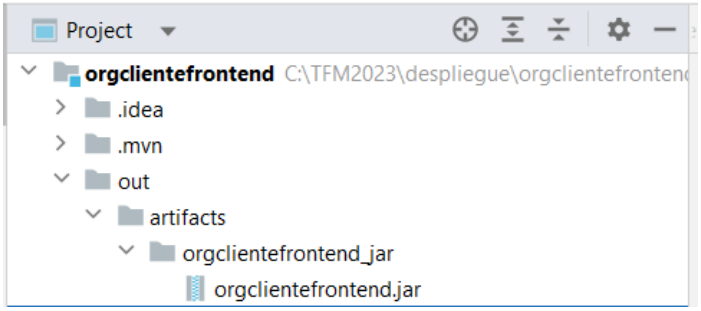

Figura 137: Ruta con el fichero jar generado (fuente elaboración propia).

2.- Creación de una imagen .

Para crear una imagen con el proyecto se copia el archivo jar a la ruta despliegue/orgclientebackend/target y se añade el fichero Dockerfile con el Código de la siguientes figura:

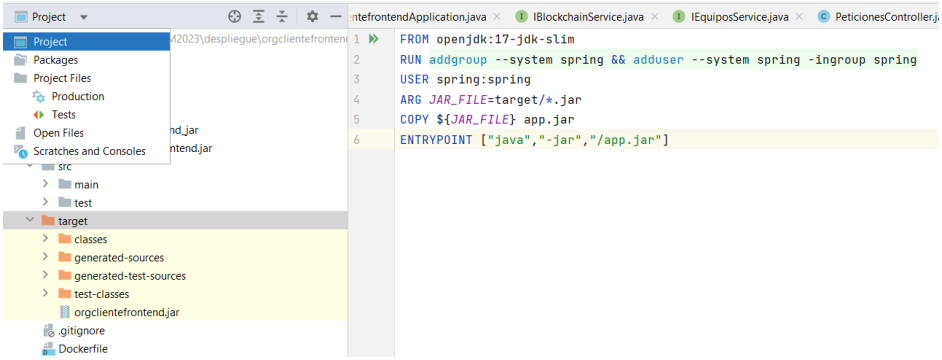

Figura 138: Ruta con los fichero Dockerfile y jar generados (fuente elaboración propia).

Por ultimo desde el directorio del proyect despliegue/orgclientebackend se ejecuta el comando que crea la imagen "docker build -t frontend/frontend_service:V1 . " .

Se puede utilizar el comando docker images para ver la imagen creada.

Figura 139: Vista de la imagen creada (fuente elaboración propia).

3.-Levantar contenedor con api rest y conectarlo a la red my-net.

Ya solo queda levantar un contenedor con la imagen creada, y conectarlo a la red my-net con el comando docker run -d --network my-net -p 18083:9001 --name frontend_container 6e333629886b (docker run --network my-net -d -p 18083:9001 --name name_container ImageID).

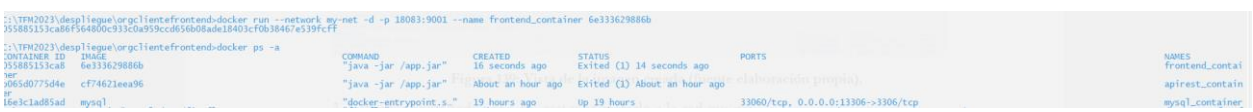

Figura 140: Vista de los tres contenedores en red (fuente elaboración propia).

1.10. Resumen y Conclusiones.

En este trabajo se ha realizado un sistema de registro de peticiones de mantenimiento utilizando tecnología de bases de datos SQL como MySQL, a la que se accede a través de una aplicación desarrollada en lenguaje Java, apoyada por el framework Spring Boot.

Además, se ha desarrollado un cliente web cuyas páginas web se han diseñado utilizando como apoyo BootStrap y el framework Spring Boot.

El sistema permite interactuar al usuario con la página web, que envía o recibe información a través de las capas del patrón DAO y la lógica del controlador, que permite comunicarse con la api rest del backend. Este, a su vez dispone de un controlador que conecta a través de las capas de su patrón DAO con la base de datos para obtener la información solicitada en función de las acciones del usuario.

Finalmente, se ha conseguido un sistema en funcionamiento diseñado para cubrir las necesidades de la organización y centralización de la información.

Además, se ha realizado una investigación práctica sobre cómo será posible integrar dicho sistema en una blockchain compartida con terceras empresas utilizando web services de Amazon (Amazon Managed Blockchain y otros) o Kaleido. El estado actual de dichos servicios, en lo que se refiere a la tecnología blockchain, no abstraen en su totalidad al usuario de la necesidad de gestionar la red y sus recursos, pero sí logran evitarle un gran trabajo de configuración de la red. La dificultad de la construcción y configuración de la infraestructura de la red y su mantenimiento desaconsejan esta opción si no se dispone de personal especializado en esta herramienta.

Asimismo, se ha desarrollado una aplicación en local como prueba de concepto, utilizando la herramienta Hyperledger Firefly que sí logra abstraer al desarrollador de la gestión de la red blockchain. En la actualidad, absolutamente recomendable utilizar esta opción. El desarrollador se olvida de la gestión de la red y sólo se ocupa del desarrollo de los smart contracts y la aplicación que utiliza la red blockchain.

Desde el punto de vista del autor, la tecnología blockchain aplicada a la auditoría de algún proceso es una revolución. Es posible marcar cualquier activo y aplicar cambios sobre él para luego poder realizar su seguimiento, garantizando la imposibilidad de alteración del registro a lo largo de tiempo y el orden en que sucedieron los cambios. Además, con absoluta transparencia para los miembros de la red. Las aplicaciones son impresionantes, desde conocer todo el mantenimiento realizado sobre un componente aeronaútico, a veces con vida útil de hasta 40 años y miles de horas de vuelo, auditar activos comprados o vendidos por determinado fondo de inversión e incluso sería posible conocer de forma individual en qué se acabó gastando el gobierno los impuestos recaudados a determinada persona.

Otra ventaja importante en opinión del autor, es que los smart contract, que controlan las transacciones en la red, podrían actuar como "contratos legales". Al fin y al cabo codifican los acuerdos a los que llegaron los miembros de la red en sus intercambios, que pueden ser económicos. Un centro de reparación aplica una tarea de mantenimiento a un equipo y cambio recibe una transferencia de dinero. Automáticamente

incluso. Por lo tanto, reduciría una parte importante de la burocracia de las organizaciones encargada de estas operaciones. Esta última aplicación no parece tan fácil que acabe utilizándose. Al fin y al cabo, cualquier persona puede comprar a otra una vivienda por un importe de dinero 10 años su sueldo, por ejemplo, con un sistema totalmente documental y, en la actualidad, poco informatizado. Y sin embargo no puede comprar acciones de valor en mercado céntimos cada una, a otra persona. Tiene que hacerlo a través de un mercado informatizado al 100% donde se negocian (transaccionan) cada día miles de millones. Esto no deja de ser curioso, aunque sea conocida la tendencia del ser humano por cuestiones históricas a utilizar un soporte físico para la persistencia, lectura o transmisión de información.

Sin embargo, aún no es fácil la implementación de esta tecnología. Cada proveedor de servicios permite construir redes blockchain de una forma. No es estándar su construcción y gestión. No es estándar la gestión de las transacciones. Hyperledger Firefly es un paso en la dirección correcta, un gran paso, proporciona una plataforma de pruebas en local y estandariza un API REST común a todas las aplicaciones para el acceso a la gestión de las transacciones de la red.

Este trabajo de fin de máster ha sido posible por los conocimientos adquiridos en el modelado y diseño bases de datos, diseño de interfaces de usuario, usabilidad, maquetado y diseño de páginas web, uso de frameworks como spring boot o Bootstrap y configuración y despliegues de sistemas sobre redes privadas o públicas.

1.11. Futuras Líneas de Trabajo.

A lo largo del proyecto se han recopilado algunas posibles mejoras.

Añadir la posibilidad de anexar documentos a las peticiones de mantenimiento como fotos de los embalajes, facturas o albaranes escaneados como comprobante documental de los transportes realizados.

Ampliar las búsquedas por otros campos. Si se comete un error puede ser difícil encontrar información.

Posibilitar la instalación de la base de datos en un servidor distinto de la lógica que la comunica con el cliente web para facilitar su escalado, disponibilidad y confiabilidad. Eso facilitaría su despliegue en internet.

Otra mejora importante es la implantación de un sistema de mensajería entre el cliente y el centro reparador, que utilice las funcionalidades que Hyperledger FireFly ofrece, para sustituir los contactos iniciales a través de medios tradicionales, por un sistema de negociación seguro e inalterable que dote de mayor sencillez, seguridad y confianza a ambas partes.

La seguridad de acceso a la aplicación se ha confiado al acceso físico al ordenador del usuario y su seguridad de acceso a la red. Implantar una capa de seguridad utilizando tecnología JWT es una mejora interesante.

Conectar el sistema a una red blockchain compartida con los centros reparadores que se unan, para poder realizar un seguimiento de las vicisitudes de los equipos cuando están en los centros reparadores. Es posible realizarlo utilizando tecnología de Amazon Web Services o Kaleido para alojar una red blockchain.

BIBLIOGRAFIA

Adhikari, B. (24 de 08 de 2020). *aws.amazon.com*. Obtenido de Building a blockchain application in Java using Amazon Managed Blockchain: https://aws.amazon.com/es/blogs/database/building-a-blockchain-application-in-java-using-amazon-managed-blockchain/

Amazon Web Service. (10 de 08 de 2020a). *Amazon Web Service*. Obtenido de https://workshops.aws: https://catalog.us-east-1.prod.workshops.aws/workshops/ce1e960e-a811-475f-a221-2afcf57e386a/en-US/01-create-a-blockchain-network

Amazon Web Service. (14 de 09 de 2022e). *Amazon Web Service* . Obtenido de Tipos de instancias de Amazon EC2: https://aws.amazon.com/es/ec2/instance-types/?trk=277c6422-fcb4-4aa1-9d80-9c5494aa7122&sc_channel=ps&s_kwcid=AL!4422!3!588732065259!e!!g!!amazon%20ec2&ef_id=Cj0KCQjw94WZBhDtARIsAKxWG-9VOenvL_P1REhry9V9oXLYolqeni0dsB3-Vpps4N4ClFN-MgZbRbYaAm2JEALw_wcB:G:s&s

Amazon Web Services. (21 de 10 de 2020b). *AWS Workshops*. Obtenido de https://workshops.aws/: https://catalog.us-east-1.prod.workshops.aws/workshops/ce1e960e-a811-475f-a221-2afcf57e386a/en-US

Amazon Web Services. (20 de 08 de 2020c). *Amzon Web Service*. Obtenido de https://workshops.aws/: https://catalog.us-east-1.prod.workshops.aws/workshops/ce1e960e-a811-475f-a221-2afcf57e386a/en-US/01-create-a-blockchain-network

Amazon Web Services. (20 de 08 de 2020d). *Amazon Web Services*. Obtenido de AWS Workshops: https://catalog.us-east-1.prod.workshops.aws/workshops/ce1e960e-a811-475f-a221-2afcf57e386a/en-US/02-set-up-a-fabric-client

Amazon Web Services. (20 de 08 de 2020e). *Amazon WebServices*. Obtenido de aws.workshops.com: https://catalog.us-east-1.prod.workshops.aws/workshops/ce1e960e-a811-475f-a221-2afcf57e386a/en-US/04-invoking-chaincode-via-api

Androulaki, E. (01 de 01 de 2020). *Hyperledger Fabric*. Obtenido de Introduccion: https://hyperledger-fabric.readthedocs.io/es/latest/whatis.html

Arlow, J. y. (2002). *UML and the Unified Process*. Addison-Wesley.

authors, T. g.-e. (01 de 01 de 2023). *Go-Ethereum*. Obtenido de Go-Ethereum Coroporativo: https://geth.ethereum.org/

BBC News Mundo. (3 de 8 de 2020). *BBC News Mundo*. Obtenido de BBC New Mundo: https://www.bbc.com/mundo/noticias-53636719

BBVA. (10 de 09 de 2022a). *bbva.com*. Obtenido de bbva.com: https://www.bbva.com/es/claves-para-entender-la-tecnologia-blockchain/

Beck, K. y. (01 de 01 de 2001). *Manifesto for Agile Software Development*. Obtenido de Manifiesto por el Desarrollo Ágil de Software: http://agilemanifesto.org/iso/es/manifesto.html

Bla, C. (28 de 04 de 2021). *https://smilecomunicacion.com/*. Obtenido de diferencias Bootstrap 4 vs 5: https://smilecomunicacion.com/diferencias-entre-bootstrap-4-y-bootstrap-5/

Bootstrap. (13 de 09 de 2022a). *Bootstrap.com*. Obtenido de Bootstrap.com: https://getbootstrap.com/docs/4.0/layout/grid/

BootStrap. (13 de 09 de 2022b). *getbootstrap.com*. Obtenido de getbootstrap.com: https://getbootstrap.com/

BootStrap. (03 de 10 de 2022c). *getbootstrap.com*. Obtenido de History: https://getbootstrap.com/docs/4.0/about/history/

BootStrap. (05 de 10 de 2022d). *getbootstrap*. Obtenido de versiones: https://getbootstrap.com/docs/versions/

Cambridge University Press. (24 de 09 de 2022). *Dictionary Cambridge*. Obtenido de https://dictionary.cambridge.org/es/: https://dictionary.cambridge.org/es/diccionario/ingles/back-office

De Tiago, d. J.-T.-S. (15 de 06 de 2013). *Cliente- Servidor*. Obtenido de wikipedia: https://es.wikipedia.org/wiki/Cliente-servidor#/media/Archivo:Cliente-Servidor.png

DirigentesDigital. (15 de 05 de 2015). *DirigentesDigital.com*. Obtenido de Dirigentes Digital: https://dirigentesdigital.com/hemeroteca/el_outsourcing-_un_sector_en_auge-FUDD13430

Encyclopaedia Brytannica. (01 de 09 de 2022). *britannica*. Obtenido de Java. Computer Programing Languaje: https://www.britannica.com/technology/Java-computer-programming-language

Ethereum. (04 de 09 de 2023). *Integraciones de Solidity disponibles*. Obtenido de https://solidity-es.readthedocs.io/es/: https://solidity-es.readthedocs.io/es/latest/

Ethereum Team. (31 de 08 de 2023). *Introducción a los contratos inteligentes*. Obtenido de https://ethereum.org/es: https://ethereum.org/es/smart-contracts/

Evolus. (01 de 01 de 2023). *Pencil Project*. Obtenido de An open-source GUI prototyping tool that's available for ALL platforms.: https://pencil.evolus.vn/

Fazt. (06 de 09 de 2021). *Solidity CRUD, Tu primer Smart Contract en Remix IDE*. Obtenido de canal Fazt Code: https://www.youtube.com/watch?v=JP-dzoDmJFw

Fenández, Y. (6 de 10 de 2017). *www.xataka.com*. Obtenido de La historia de las hojas de cálculo digitales: de idea descartada a herramienta imprescindible: https://www.xataka.com/historia-tecnologica/la-historia-de-las-hojas-de-calculo-digitales-de-idea-descartada-a-herramienta-imprescindible

Github Hyperledger FireFly Ethconnect. (10 de 07 de 2023). Obtenido de Documentación : https://github.com/hyperledger/firefly-ethconnect/tree/v3.2.11

Guzmán, A. J. (22 de 08 de 2022). *Udemy*. Obtenido de Programación Reactiva con Spring Boot 2 y Spring WebFlux: https://www.udemy.com/course/programacion-reactiva-con-spring-webflux-reactor/

Hyperledger. (28 de 09 de 2021). Obtenido de Hypeledger Fundation Blog: https://www.hyperledger.org/blog/2021/09/28/introducing-hyperledger-firefly-a-multi-party-system-for-enterprise-data-flows

Hyperledger. (11 de 07 de 2023a). *Documentación FireFly*. Obtenido de Github FireFly: https://hyperledger.github.io/firefly/overview/key_features.html

Hyperledger. (11 de 07 de 2023b). *Huperledger Foundation*. Obtenido de Huperledger FireFly: https://hyperledger.github.io/firefly/v1.2.0/overview/multiparty_features.html

Hyperledger. (01 de 01 de 2023c). *Hyperledger FireFly*. Obtenido de Firefly MTLS Data Exchange: https://github.com/hyperledger/firefly-dataexchange-https

Hyperledger. (06 de 09 de 2023d). *Introduction to Supernodes*. Obtenido de Hyperledger FireFly: https://hyperledger.github.io/firefly/v1.0.4/overview/firefly_supernode.html

HyperLedger. (06 de 09 de 2023e). *Enterprise multi-party systems*. Obtenido de HyperLedger FireFly: https://hyperledger.github.io/firefly/v1.0.4/overview/multiparty.html

IBM. (10 de 09 de 2022a). *ibm.com*. Obtenido de ibm.com: https://www.ibm.com/es-es/topics/what-is-blockchain

IBM. (10 de 09 de 2022b). *ibm.com*. Obtenido de ibm.com: https://www.ibm.com/topics/blockchain-security

IPFS. (23 de 07 de 2023). *https://ipfs.tech/*. Obtenido de IPFS powers the Distributed Web: https://ipfs.tech/

Jacobson, I. B. (1999). *The Unified Software Development Process.* Addison-Wesley.

Jetbrains. (05 de 10 de 2020). *jetbrains.com*. Obtenido de Un recorrido nostálgico: https://www.jetbrains.com/es-es/lp/intellijidea-20-anniversary/

jQuery. (05 de 10 de 2022). *jquery.com/*. Obtenido de Portal, write less, do more: https://jquery.com/

Kaleido. (29 de 06 de 2022a). Obtenido de Build Sustainable Supply Chains with Hyperledger FireFly: https://youtu.be/pGnYlUW5aLI

Kaleido. (06 de 09 de 2022b). *Build Sustainable Supply Chains with Hyperledger FireFly* . Obtenido de Youtube Kaleido Channel: https://www.youtube.com/watch?v=pGnYlUW5aLI&t=300s

Kaleido. (01 de 07 de 2022c). *Cómo construir una solución blockchain con FireFly Hyperledger*. Obtenido de Kaleido youtube: https://youtu.be/DkuyhyQF_Tk

Kaleido. (10 de 07 de 2023). *Document Exchange*. Obtenido de Blockchain Platform: https://www.kaleido.io/blockchain-platform/document-exchange

Kaleido Documentacion Desarrolladores. (23 de 07 de 2023). Obtenido de Kaleido: https://docs.kaleido.io/developers/app/event-streams/

Lantolin. (06 de 09 de 2020). *Proceso Unificado*. Obtenido de wikipedia: https://es.wikipedia.org/wiki/Proceso_unificado#/media/Archivo:Fases_y_Flujos_de_trabajo_en_PUR.svg

MongoDB. (07 de 10 de 2022a). *MongoDB*. Obtenido de Empresa: https://www.mongodb.com/es/companWeby

MongoDB. (17 de 10 de 2022b). *Compass*. . Obtenido de The GUI for MongoDB: https://www.mongodb.com/products/compass

Mozilla Contributors, M. (18 de 07 de 2023a). *Que es Javascript*. Obtenido de Fundamentos de Javscript: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/JavaScript_basics

Mozilla Contributors, M. (24 de 07 de 2023b). *JavaScript*. Obtenido de MDN Desarrolladores: https://developer.mozilla.org/es/docs/Web/JavaScript

Mozilla Contributors, M. (02 de 08 de 2023c). *What is Javascript?* Obtenido de https://developer.mozilla.org/es/docs/Learn/JavaScript: https://developer.mozilla.org/es/docs/Learn/JavaScript/First_steps/What_is_JavaScript

New World Enciylopedia. (24 de 09 de 2022). *Research begins here New World encyclopedia*. Obtenido de https://www.newworldencyclopedia.org: https://www.newworldencyclopedia.org/entry/Interoperability

Oracle. (04 de 09 de 2023a). *MySQL Community Downloads*. Obtenido de https://dev.mysql.com/: https://dev.mysql.com/downloads/

Oracle. (04 de 09 de 2023b). *MySql Products*. Obtenido de https://www.mysql.com/: https://www.mysql.com/products/

Oracle. (04 de 09 de 2023c). *MySQL Workbench Manual*. Obtenido de https://dev.mysql.com/: https://dev.mysql.com/doc/workbench/en/wb-intro.html

Pabex. (23 de 09 de 2021). *Blockchain con Hyperledger Fabric: de la teoría a la práctica.* . Obtenido de https://www.youtube.com/watch?v=PNo_F49tHZ4

Portal administración electrónica . (24 de 09 de 2022). *Portal administración electrónica Gobierno de España*. Obtenido de https://administracionelectronica.gob.es: https://www.newworldencyclopedia.org/entry/Interoperability

Pressman, R. S. (2010b). 2.5.2 Fases del proceso unificado. En R. S. Pressman, *Ingenieria del Software. Un enfoque práctico. Séptima Edición.* (págs. 45-48). México D.F.: Mc Graw Hill.

Pressman, R. S. (2010c). 5.1 Ingenieria de Requerimeintos.Concepción. En R. S. Pressman, *Ingeniería de Software. Un Enfoque práctico. Séptima Edición* (págs. 102-103). Mexico D. F.: Mc Graw Hill.

Pressman, R. S. (2010d). 3.5.8 El proceso unificado ágil. En R. S. Pressman, *Ingeniería de Software. Un enfoque práctico. Séptima Edición.* (págs. 75-76). México D.F.: Mc Graw Hill.

Pressman, R. S. (2010e). 5.3.2 Despliegue de la función de calidad. En R. S. Pressman, *Ingeniería del Software. Un enfoque práctico.Séptima Edición.* (pág. 111). México D.F.: Mc Graw Hill.

Pressman, R. S. (2010f). 5.4 Desarrollo de casos de uso. En R. S. Pressman, *Ingeniería de Software. Un enforque práctico. Séptima edición* (pág. 113). Méxco D. F.: Mc Graw Hill.

Pressman, R. S. (2010g). 7.5.4 Modelo de contenido de las webapps. En R. S. Pressman, *Ingeniería de Software. Un enforque práctico. Séptima edición* (págs. 176-177). México D,F.: Mc Graw Hill.

Pressman, R. S. (2010h). 17.7.2 Arquitectura de las webapps. En R. S. Pressman, *Ingeniería de Software. Un enforque práctico. Séptima edición* (págs. 328-329). Mc Graw Hill.

Real Academia de la Lengua y otros. (24 de 09 de 2022b). *DEJ Panhispánico*. Obtenido de https://dpej.rae.es/: https://dpej.rae.es/lema/outsourcing

Real Academia Española de la Lengua y otros. (24 de 09 de 2022a). *DEJ Panhispánico*. Obtenido de https://dpej.rae.es: https://dpej.rae.es/lema/back-office

Solidity. (03 de 09 de 2023). *Instalando Solidity*. Obtenido de https://solidity-es.readthedocs.io: https://solidity-es.readthedocs.io/es/latest/installing-solidity.html

Solidity Team. (04 de 09 de 2023). *About Solidity*. Obtenido de https://soliditylang.org/: https://soliditylang.org/about/

Studio 3T. (17 de 10 de 2022). *Studio 3T free*. Obtenido de Download: https://studio3t.com/download-studio3t-free

thymeleaf. (12 de 11 de 2020). *www.thymeleaf.org*. Obtenido de Spring MVC view layer: Thymeleaf vs. JSP: https://web.archive.org/web/20130803035747/http://www.thymeleaf.org/thvsjsp.html

VanillaJS. (09 de 10 de 2022). *vanilla-js.com*. Obtenido de Portal. : http://vanilla-js.com/