

Universidad Internacional de La Rioja (UNIR)

**Escuela Superior de Ingeniería y
Tecnología**

Grado en Ingeniería Informática

Petchain: DApp de gestión de mascotas

Ubicación del código fuente:

https://github.com/cescari/PetChain_TFG

Trabajo Fin de Grado

Presentado por: Escario Bajo, Carlos

Director/a: Martínez Muñoz, Miriam.

Ciudad: San Lorenzo de El Escorial (Madrid)

Fecha: 05/06/2019

Por y para Carolina.

Resumen

El presente trabajo es una muestra de cómo es posible la realización de proyectos de gestión con Smartcontracts dentro de un ámbito corporativo o administrativo. La aplicación, bautizada como **Petchain**, pretende mostrar cómo es posible salvar la falta de comunicación existente en la actualidad, debido a la ausencia de interconexión entre los sistemas informáticos dentro del sector veterinario nacional, aprovechando las capacidades de confiabilidad e inmutabilidad que ofrece una red Blockchain y el sistema de ficheros IPFS.

Petchain es una aplicación "web clásica", que permite la gestión de identidades de mascotas domésticas. Desarrollada como una Blockchain bajo una red *Ethereum*, funcionando en un entorno local en un servidor *Ganache* y con un frontal HTML, permite la gestión y administración de la información relacionada con las mascotas, los datos de sus dueños así como la distribución de los mismos, entre los profesionales veterinarios.

Palabras clave: Blockchain, web, Ethereum, Ganache, Truffle, Vue.js token, nodo, IPFS

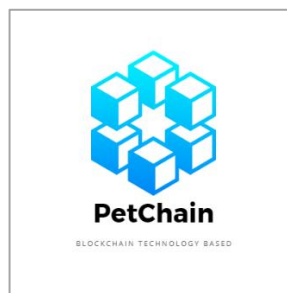


Ilustración 1: Petchain logo

Summary

The current project is an example of how it is possible to carry out management projects with Smartcontracts within a corporate or administrative environment. The application, called **Petchain**, is intended to show how it is possible to overcome the today's lack of communication, due to the absence of interconnection among systems within the national veterinary sector, taking advantage of the capabilities, reliability and immutability offered by a Blockchain network and the IPFS file system.

Petchain is a "classic web" project, which allows the management of domestic pet identities. Developed as a Blockchain under an Ethereum network, working in a local environment on a *Ganache* server and with an HTML front end, it allows the management and administration of information regarding to pets, their owners' data as well as their distribution among veterinary professionals.

Keywords: Blockchain, web, Ethereum, Ganaché, Truffle, Vue.js, token, node, IPFS.

Índice

| | |
|--|-----------|
| Resumen | 1 |
| Summary | 1 |
| Índice | 2 |
| Índice de ilustraciones | 5 |
| Índice de Tablas | 7 |
| 1 Introducción | 9 |
| 1.1 Presentación del problema..... | 9 |
| 1.2 Motivación..... | 10 |
| 1.3 Objetivos..... | 12 |
| 1.3.1 Objetivo general..... | 12 |
| 1.3.2 Objetivos específicos | 12 |
| 2 Estado del arte. | 14 |
| 2.1 La Blockchain..... | 14 |
| 2.1.1 Blockchain públicas..... | 14 |
| 2.1.2 Blockchain privadas | 15 |
| 2.1.3 Blockchain de consorcio..... | 15 |
| 2.2 Características | 16 |
| 2.2.1 ¿Qué es un bloque?..... | 16 |
| 2.2.2 Proof of work..... | 19 |
| 2.2.3 Proof of stake..... | 20 |
| 2.3 IPFS..... | 20 |
| 2.4 DApp. ¿Qué es? | 21 |
| 2.5 Smartcontracts..... | 21 |
| 3 Petchain. Especificación de requisitos. | 22 |
| 3.1 Propósito del documento..... | 22 |
| 3.2 Alcance del documento | 22 |
| 3.3 Participantes en el proyecto | 22 |
| 3.3.1 Presentación de participantes | 22 |
| 3.4 Organizaciones involucradas | 24 |
| 3.5 Descripción del sistema | 24 |
| 3.5.1 Perspectiva del producto..... | 24 |
| 3.5.2 Funciones del producto..... | 24 |
| 3.5.3 Características del usuario..... | 25 |
| 3.5.4 Restricciones | 25 |
| 3.6 Descripción técnica del sistema | 25 |

| | | |
|----------|---|-----------|
| 3.6.1 | Front-End | 25 |
| 3.6.2 | Seguridad | 26 |
| 3.6.3 | Back-End | 26 |
| 3.6.4 | Entorno de ejecución | 26 |
| 3.7 | Objetivos del sistema | 27 |
| 3.8 | Catálogo de requisitos del sistema..... | 31 |
| 3.8.1 | Requisitos de información | 31 |
| 3.8.2 | Requisitos funcionales | 33 |
| 3.8.3 | Requisitos no funcionales | 38 |
| 3.9 | Matriz de rastreabilidad | 43 |
| 3.9.1 | Matriz de rastreabilidad | 43 |
| 3.10 | Casos de uso | 44 |
| 3.10.1 | Gestión de mascotas | 44 |
| 3.10.2 | Modificación de propietarios..... | 46 |
| 3.10.3 | Consulta de saldo | 47 |
| 3.10.4 | Consulta de transferencias..... | 48 |
| 3.10.5 | Registro en la Blockchain..... | 49 |
| 4 | Descripción técnica. | 50 |
| 4.1 | Arquitectura..... | 51 |
| 4.1.1 | Modelo | 52 |
| 4.1.2 | Vista..... | 52 |
| 4.1.3 | Controlador | 53 |
| 4.1.4 | Análisis de elementos: Componentes y Vistas. | 53 |
| 4.2 | Arquitectura técnica. | 55 |
| 4.3 | Arquitectura del backend..... | 57 |
| 4.3.1 | Almacenamiento en la IPFS..... | 57 |
| 4.3.2 | Recuperación de información..... | 59 |
| 4.4 | Entorno de desarrollo | 60 |
| 4.4.1 | Node.js..... | 60 |
| 4.4.2 | Vue.js..... | 61 |
| 4.4.3 | Truffle..... | 62 |
| 4.4.4 | Ganache | 63 |
| 4.4.5 | Web3.js..... | 64 |
| 4.4.6 | Bootstrap | 65 |
| 4.5 | Herramientas utilizadas..... | 66 |
| 4.5.1 | Argo UML..... | 66 |
| 4.5.2 | REM..... | 66 |
| 4.5.3 | GitHub Desktop..... | 66 |
| 4.5.4 | Trello..... | 66 |

| | | |
|-----------|--|-----------|
| 4.5.5 | Visual Studio Code..... | 66 |
| 5 | Petchain. Descripción funcional | 68 |
| 5.1 | Login | 68 |
| 5.1.1 | Registro en la Blockchain..... | 69 |
| 5.2 | Pantalla principal..... | 70 |
| 5.2.1 | Menú de navegación..... | 71 |
| 5.2.2 | Footer (pie de pantalla) | 71 |
| 5.3 | Alta de mascotas..... | 72 |
| 5.4 | Modificación de mascotas | 72 |
| 5.5 | Baja de mascotas..... | 73 |
| 5.6 | Transferencias y saldo de ETHERS..... | 74 |
| 6 | Test de código..... | 76 |
| 6.1 | Validación de código | 76 |
| 6.1.1 | Validación inicial (*.html) | 76 |
| 6.1.2 | Validación inicial (*.css)..... | 78 |
| 6.2 | Test Smartcontracts | 80 |
| 6.2.1 | Test JoinedUser.sol | 80 |
| 6.2.2 | Test PetChain.sol..... | 81 |
| 6.2.3 | Test registerTX | 82 |
| 6.2.4 | Resultados de los test..... | 82 |
| 6.3 | Validación de objetos JSON..... | 82 |
| 6.4 | Visualización en diferentes dispositivos | 83 |
| 6.4.1 | Realización de pruebas..... | 83 |
| 6.4.2 | Test Smartphone..... | 84 |
| 6.4.3 | Test Tablet..... | 85 |
| 7 | Conclusiones y trabajo futuro..... | 86 |
| 7.1 | Análisis sobre objetivos establecidos | 86 |
| 7.2 | Trabajo futuro..... | 87 |
| 8 | Bibliografía | 89 |
| 9 | Glosario de términos | 90 |
| 10 | Referencias..... | 93 |

Índice de ilustraciones

| | |
|--|----|
| Ilustración 1: Petchain logo | 1 |
| Ilustración 2: Reto ECO Asturias | 11 |
| Ilustración 3: Logotipos de Bitcoin, Ether y Monero | 15 |
| Ilustración 4: Representación de la cadena de bloques..... | 17 |
| Ilustración 5: Ejemplo de bloque genesis.json | 17 |
| Ilustración 6: Bloque Genesis | 18 |
| Ilustración 7: mempool | 19 |
| Ilustración 8: Esquema figurado DApp | 21 |
| Ilustración 9: Gestión de mascotas..... | 44 |
| Ilustración 10: Caso de uso modificación de propietarios | 47 |
| Ilustración 11: Consulta del saldo en Ethers..... | 47 |
| Ilustración 12: Caso de uso consulta de transferencias en ETH..... | 49 |
| Ilustración 13: Caso de uso de registro en la Blockchain..... | 49 |
| Ilustración 14: Esquema gráfico de herramientas de Petchain | 50 |
| Ilustración 15: Patrón MVC..... | 51 |
| Ilustración 16: Estructura de los ficheros del Modelo..... | 52 |
| Ilustración 17: Estructura de ficheros de la Vista | 52 |
| Ilustración 18: Esquema de la capa controladora | 53 |
| Ilustración 19: Elementos de un componente Vue.js | 54 |
| Ilustración 20: Flujo de composición de la vista.Contactoview.vue | 55 |
| Ilustración 21: Esquema de componentes..... | 56 |
| Ilustración 22: Esquema de generación y almacenamiento de la información. | 59 |
| Ilustración 23: Esquema de recuperación de la información de la IPFS | 60 |
| Ilustración 24: Comando de instalación de Truffle | 63 |
| Ilustración 25: Comando de inicialización de un proyecto Truffle..... | 63 |
| Ilustración 26: Web de Truffle framework. | 64 |
| Ilustración 27: Pantalla principal de PetChain..... | 68 |
| Ilustración 28: Login PetChain..... | 68 |
| Ilustración 29: Login incorrecto..... | 69 |
| Ilustración 30: Ventana recordatorio de contraseña..... | 69 |
| Ilustración 31: Ventana de registro en la Blockchain. | 70 |
| Ilustración 32: Opciones de navegación | 70 |
| Ilustración 33:Pantalla de Alta de mascotas | 72 |
| Ilustración 34: Ventana de aviso/error | 72 |

| | |
|--|----|
| Ilustración 35: Ventanas de confirmación y finalización..... | 72 |
| Ilustración 36: Pantalla de Modificación de mascotas..... | 73 |
| Ilustración 37: Pantalla de Baja de mascotas | 73 |
| Ilustración 38: Confirmación de la Baja de una mascota. | 74 |
| Ilustración 39: Tranferencias de ETH | 74 |
| Ilustración 40: : Resultados W3C de la validación de index.html | 77 |
| Ilustración 41: Validación W3C index.html..... | 78 |
| Ilustración 42: Resultados W3C de main.css..... | 79 |
| Ilustración 43: Resultado final del W3C de la validación de main.css | 79 |
| Ilustración 44: remix.ethereum.org IDE | 80 |
| Ilustración 45: Simulación de LG Optimus L70 | 84 |
| Ilustración 46: Simulación de Iphone7 | 84 |
| Ilustración 47: Simulación iPhone7 en horizontal | 85 |
| Ilustración 48: Simulación de iPad..... | 85 |
| Ilustración 49: Evolución del desarrollo de DApps | 87 |

Índice de Tablas

| | |
|--|----|
| Tabla 1: Tipos de blockchain | 14 |
| Tabla 2: Dirección del proyecto | 23 |
| Tabla 3: Analista y desarrollador del proyecto | 23 |
| Tabla 4: Colaborador del proyecto | 23 |
| Tabla 5: Dirección facultativa y Veterinaria | 23 |
| Tabla 6: Organización 1 | 24 |
| Tabla 7: Organización 2 | 24 |
| Tabla 8: Objetivo de Alta de mascotas | 27 |
| Tabla 9: Objetivo Alta de dueños | 27 |
| Tabla 10: Objetivo de Modificación de datos | 28 |
| Tabla 11: Objetivo de Baja de mascotas | 28 |
| Tabla 12: Objetivo de Consulta de datos de una mascota | 28 |
| Tabla 13: Objetivo de Consulta de saldo | 29 |
| Tabla 14: Objetivo de Control de acceso | 29 |
| Tabla 15: Objetivos de visualización | 30 |
| Tabla 16: Objetivo de la navegación de la aplicación | 30 |
| Tabla 17: Objetivo de traspaso de ETH | 30 |
| Tabla 18: Requisito de Información del veterinario | 31 |
| Tabla 19: Requisitos de información de mascotas | 32 |
| Tabla 20: Requisitos de información de propietarios | 33 |
| Tabla 21: Requisitos de información del saldo en ETH | 33 |
| Tabla 22: Requisito funcional de acceso al sistema | 34 |
| Tabla 23: Requisito funcional de registro en la Blockchain | 34 |
| Tabla 24: Requisito funcional de identificación de veterinarios | 35 |
| Tabla 25: Requisito funcional de alta de una mascota | 35 |
| Tabla 26: Requisito funcional de alta de un propietario | 36 |
| Tabla 27: Requisito funcional de modificación de datos de una mascota | 36 |
| Tabla 28: Requisito funcional para la baja de una mascota | 37 |
| Tabla 29: Requisito funcional para la visualización de weis/ETH traspasado | 37 |
| Tabla 30: Requisito funcional Traspaso de ETH o weis | 38 |
| Tabla 31: Requisito funcional para consultar datos | 38 |
| Tabla 32: Requisito no funcional de visualización de Petchain | 39 |
| Tabla 33: Requisito no funcional del pago en weis | 39 |
| Tabla 34: Requisito no funcional del tiempo de sesión | 40 |
| Tabla 35: Requisito no funcional de usuarios autorizados | 40 |

| | |
|--|----|
| Tabla 36: Requisito no funcional del nº de cuentas en Ganache | 40 |
| Tabla 37: Requisito no funcional de campos obligatorios | 41 |
| Tabla 38: Requisito no funcional de definición de opciones de navegación..... | 41 |
| Tabla 39: Requisito no funcional de carga de datos estáticos. | 42 |
| Tabla 40: Requisito no funcional de obligación de validar el código. | 42 |
| Tabla 41: Requisito no funcional de confirmación de transacciones..... | 42 |
| Tabla 42: Matriz de rastreabilidad Objetivos-Req. Funcionales-Req. no Funcionales- | 43 |
| Tabla 43: Ficha de caso de uso Alta de mascotas..... | 44 |
| Tabla 44: Ficha del caso de uso alta de propietario..... | 45 |
| Tabla 45: Ficha del caso de uso de modificación de mascotas | 45 |
| Tabla 46: Ficha del caso de uso de baja de mascotas. | 46 |
| Tabla 47: Ficha del caso de uso consulta de mascotas..... | 46 |
| Tabla 48: Ficha del caso de uso modificación de propietarios..... | 47 |
| Tabla 49: Ficha del caso de uso consulta de saldo | 48 |
| Tabla 50: Ficha del caso de uso consulta de transferencias..... | 48 |
| Tabla 51: Ficha del caso de uso registro en la Blockchain. | 49 |
| Tabla 52: Relación Arquitectura - Tecnología..... | 51 |
| Tabla 53: Comando de instalación de Vue-CLI | 62 |
| Tabla 54: Instalación de web3.js | 65 |
| Tabla 55: Resultados de la validación inicial de los ficheros html | 76 |
| Tabla 56: Resultados del código html validado..... | 78 |
| Tabla 57: Resultados de la validación de main.css | 79 |
| Tabla 58: Resultado de la validación de main.css | 79 |
| Tabla 59: Descripción de JoinedUser.sol | 80 |
| Tabla 60: Ejecución de test sobre JoinedUser.sol | 81 |
| Tabla 61: Descripción de PetChain.sol..... | 81 |
| Tabla 62: Ejecución de test sobre PetChain.sol | 81 |
| Tabla 63: Descripción de registerTX.sol | 82 |
| Tabla 64: Ejecución de test sobre registerTX.sol..... | 82 |
| Tabla 65: Resultados validación de ficheros JSON | 83 |

1 Introducción

¿Por qué Blockchain? ¿Si ya tiene más de 20 años, qué trae nuevo? ¿Por qué he de fiarme de algo que se califica a sí mismo como inmutable? ¿Pero es seguro? En la era de las APIs, ¿por qué debo de fiarme de un SmartContract? Pero y eso de un SmartContract, ¿qué es? ¿Necesito cambiar las competencias del equipo de desarrollo? Y además, ¿debo invertir en criptomonedas para poder hacer mi aplicación empresarial? Etc...

Estas cuestiones y otras muchas, son las que cualquier analista, jefe de proyecto, responsable de TI o incluso de negocio que se enfrenta a un nuevo desarrollo, podría llegar a hacerse si le planteasen la necesidad de diseñar e implementar un proyecto en una blockchain corporativa.

En el presente trabajo, se pretende ofrecer la visión de cómo es posible desarrollar una aplicación de gestión basado en un nuevo paradigma de desarrollo: **la Blockchain**. Observemos que se ha dicho “nuevo”, pues no es el objetivo de este proyecto, bajo ningún concepto, querer transmitir el concepto de “mejor”. Por otro lado, se quiere dejar claro que no es necesario saber invertir en criptomonedas, o cual es la cotización actual del Bitcoin o el Ether para poder desarrollar una aplicación basada en una Blockchain.

1.1 Presentación del problema

En la actualidad, la movilidad de las personas y las cosas han forzado que estemos viviendo en la era de la interconexión a todos los niveles, ya sea personal, empresarial o administrativa. La mayoría de los sectores, para realizar sus actividades de negocio o de gestión, se ven obligados a consultar datos de otras fuentes. Las empresas y administraciones del territorio español, necesitan exponer y compartir sus datos para poder tener un control y ofrecer los servicios que los ciudadanos requieran. Es este flujo de información el que sirve de motor para gran parte de los sectores económicos en el mundo actual, y cuando éste se rompe, o simplemente no existe, notamos que todo se vuelve más lento e ineficiente.

Pues bien, este escenario, el de la falta de comunicación global, es el que en la actualidad se está produciendo en el sector veterinario español. Y este problema, además de los profesionales veterinarios, quien lo sufre son los usuarios del servicio, las personas y sus mascotas. ¿Pero y por qué? Pues debido a la movilidad de las personas y las familias que, junto con sus mascotas, se trasladan a otras residencias constantemente en cuestión de horas, por razones personales, de trabajo o de ocio.

En España, según datos de la web de profesionales del sector veterinario

Petshopmagazine¹, en el año 2018 había censados 5.147.000 perros y 2.265.000 gatos. Pues bien, es esta movilidad y la problemática administrativa que ella conlleva, en la que se quiere centrar el desarrollo del presente trabajo. Estos traslados familiares, ya sean ocasionales o permanentes, a nivel de mascotas también requieren de un registro y control administrativo por parte de los profesionales del sector veterinario. Éstas necesitan un control sanitario para su vacunación periódica, una atención puntual por algún tipo de enfermedad o el registro en sus nuevos domicilios. Este tipo de atención tiene como elemento angular de entrada el **Código de Identificación**, o lo que comúnmente se conoce como “**el chip**”. Es un dispositivo electrónico, de un tamaño mayor al de un grano de arroz y compuesto por una electrónica miniaturizada encapsulada en plástico, que almacena un código alfanumérico de 15 posiciones único por cada dispositivo. Es el DNI del animal, el cual permite asociarlo al dueño y dotarle de un historial de domicilio, teléfono de contacto, historial veterinario, movimientos, etc.

La problemática surge debido a la ausencia de interconexión entre los diferentes servicios sanitarios veterinarios de las comunidades autónomas en España. Según el sistema SIACYL², Sistema de Identificación Animal de Castilla y León, esta comunidad sólo tiene conexión con Castilla y La Mancha (SIACAM), Aragón (RIACA), Murcia (SIAMU) y Melilla (SIAMEL). Por ejemplo, se producen con relativa asiduidad casos como los que ocurren en alguno de los servicios veterinarios de la provincia de Ávila que, a pesar de encontrarse a escasos kilómetros de la Comunidad de Madrid, no tienen medios para localizar a los dueños de una mascota extraviada procedente de esta Comunidad mediante la consulta del chip identificativo. En estas ocasiones, es necesario realizar una llamada telefónica a los servicios veterinarios de Madrid, en los que sólo existe atención telefónica en horario laboral y de lunes a viernes, con lo que se dan circunstancias en las que si un animal se extravía el sábado por la mañana y se intenta localizar al dueño, esto no sea posible hasta el lunes siguiente y sólo mediante una llamada de telefónica. Es decir, se antoja necesario la búsqueda de una solución que permita la interconexión de los diversos sistemas sanitarios animales de las diferentes Comunidades Autónomas, con el objetivo de dar servicio tanto a las mascotas como a sus propietarios.

1.2 Motivación

En el punto anterior se ha expuesto la problemática, pero la motivación que provoca el desarrollo del presente proyecto es el de dar una solución al reto lanzado por el Consorcio


¹ <http://petshopsmagazine.com/mascotas/censo-mascotas-espana/>

² www.siacyl.org

Alastria³ en su Open Call del año 2018⁴. En esta iniciativa cada uno de sus diez Ecosistemas de Coordinación (ECO) que Alastria tiene repartidos por España, lanzaron cada uno de ellos una propuesta de desarrollo sobre Blockchain, que diese solución a una problemática en concreto.

Dentro de esta iniciativa el reto lanzado por el ECO de Asturias era el de un Registro de Animales de Compañía⁵ que proponía la dificultad que existe en el sector veterinario nacional debido a los problemas que surgen por la falta de comunicación entre las diversas Comunidades Autónomas. Por otro lado, esta problemática ya era conocida previamente a través de informaciones recabadas en el Centro Veterinario García-Ochoa de Sotillo de la Adrada (Ávila), por lo que la conjunción de ambas circunstancias dieron lugar a aceptar la idea de crear una aplicación que diese solución a los dos problemas planteados, el de comunicar los centros veterinarios de toda España y que ésta sea desarrollada en una Blockchain.

ECO ASTURIAS



Reto 1

Registro de animales de compañía: en 1 de cada 3 hogares asturianos vive un animal de compañía convenientemente censado en el registro del Principado de Asturias. Este registro autonómico no se encuentra interconectado con el del resto de comunidades autónomas, siendo el propietario del animal el responsable de realizar las gestiones oportunas con los distintos registros nacionales en caso de viaje o cambio de domicilio. Además el propietario debe poder presentar la cartilla sanitaria del animal, en formato papel, convenientemente actualizada con las vacunaciones realizadas y otros datos sanitarios.

Actualmente estos retos se resuelven con la existencia de un registro centralizado a nivel nacional y gestionado por los colegios de veterinarios donde se ponen en común los datos (identificación, vacunaciones, etc.) de las distintas bases de datos, siendo necesario realizar parte de los trámites en papel. A nivel europeo e internacional existen registros centralizados similares, donde vuelcan sus datos cada uno de los registros nacionales integrados.

La aparición de tecnologías de registro distribuido como blockchain parecen más adecuadas para resolver estos retos de agregación de datos de diferentes sistemas de manera descentralizada, inmutable, segura y accesible a nivel mundial, mientras que la incorporación de una Identidad Digital al animal permitiría gestionar de una manera más eficaz la información asociada.

¿Cómo podría mejorar Alastria la gestión de estos registros de identidad animal y datos sanitarios?
¿Qué ventajas obtendrían ciudadanos y veterinarios?

Reto 2

[Gijón con la movilidad sostenible: el plan estratégico de Gijón contempla de forma destacada la](#)

Ilustración 2: Reto ECO Asturias (Fuente:

https://drive.google.com/file/d/1JxmV0252ikfsPAwdW_mIRmED2-Ww5VK6/view)

³ <https://alastria.io>

⁴ https://medium.com/@alastria_es/open-call-alastria-2f09f6865717

⁵ https://drive.google.com/file/d/1JxmV0252ikfsPAwdW_mIRmED2-Ww5VK6/view

El desarrollo afrontado se enmarca, además de para abordar el reto propuesto, en una iniciativa exclusiva de carácter académico con el objeto de servir como base para el presente Trabajo de Final de Grado.

1.3 Objetivos

Se proponen una serie de objetivos que se pretenden conseguir al finalizar el desarrollo del presente trabajo, que se enumeran a continuación y que están ordenados en base a su importancia.

1.3.1 Objetivo general

Como se ha expuesto con anterioridad, la problemática actual en el caso de la localización de las mascotas en el ecosistema de las Comunidades Autónomas, por la falta de conexión entre los diferentes entornos, se antoja complicada. Se pretende entonces mostrar cómo el modelo, basado en una DApp y ejecutándose en una Blockchain, permite salvar esta “trinchera”, unificando en un solo sistema toda la gestión de las identidades de las mascotas

1.3.2 Objetivos específicos

1. **Objetivo 1:** Demostrar que la Blockchain y la IPFS son entornos viables al 100%, para el desarrollo de aplicaciones de gestión. No es objeto del presente trabajo, ni del proyecto asociado, el querer demostrar el concepto de “mejor”, si no el de “nuevo”, es decir, que es posible una alternativa al sistema de trabajo con bases de datos tradicionales, que aporte las características de descentralización e inmutabilidad.
2. **Objetivo 2:** Web usable, accesible desde cualquier dispositivo y atractiva. Estos tres criterios, a juicio del autor, deberían ser parte de cualquier desarrollo web. Para los dos primeros se van a establecer criterios dentro de los requisitos no funcionales que, obviamente, deberán estar cumplidos al finalizar el desarrollo. Para el tercero se seguirá el principio de “menos es más” a la hora de abordar el diseño para poder lograrlo.
3. **Objetivo 3:** Software coste cero. Se pretende hacer un desarrollo con software de uso libre, que no implique desembolso alguno a las personas o entidades que lo aborden. Esto incluye IDEs de desarrollo, frameworks, iconos, entornos de alojamiento, herramientas de validación, servidores y entornos de despliegue.
4. **Objetivo 5:** No hace falta ser un experto en criptodivisas. Demostrar que para el desarrollo de aplicaciones, y más en concreto de una DApp, no es necesario ser un

experto conocedor del mundo Bitcoin o Ether, ni dominar la cotización de ambas en el mercado de las inversiones.

5. **Objetivo 6:** Aumentar las competencias del autor en la programación de lenguajes como Javascript en entornos Node.js, Solidity para el desarrollo de Smartcontracts y el manejo del framework Web3JS para la combinación de todas estas tecnologías.

2 Estado del arte.

Para comprender el entorno en el que nos pretendemos desenvolver, es necesario comprender de manera general los elementos que la componen, haciendo hincapié en aquellos que resultan “novedosos” en el tipo de desarrollo que se pretende mostrar. Por lo tanto es necesario tener una visión general de:

- Blockchain.
- IPFS (Inter Planetary File System).
- DApp.
- Smartcontracts.

2.1 La Blockchain

¿Pero qué es Blockchain? La respuesta es, una red P2P (*peer-to-peer*) en la que se crean transacciones digitales, y que funciona con el mismo concepto que el de un libro mayor de contabilidad o *ledger*. En esta red la información se almacena en bloques, en donde cada uno de ellos contiene información relativa al bloque anterior, lo que permite ir formado una cadena con información accesible pero inmutable.

En la actualidad la blockchain ha extendido su uso a muchos sectores económicos, por lo que ahora estamos en disposición de realizar una clasificación, según su uso:

| Tipos de Blockchain | | | | | |
|---------------------|---------------------------------------|-------------|--|---------------|--|
| Publicas | | Privadas | | De consorcio | |
| Bitcoin | Sin permisos. Identidades anónimas | Hyperledger | Con permisos. Identidades conocidas | B3i (seguros) | Con permisos. Identidades conocidas |
| Ethereum | | Fabric | | Alastria | |
| Monero | | Ripple | | | |

Tabla 1: Tipos de blockchain

2.1.1 Blockchain públicas

Este modelo es el primero que surgió e Internet ha sido el entorno en donde se ha desarrollado hasta alcanzar la popularidad con la que ahora cuenta. Los tipos que podemos destacar por su popularidad serían Bitcoin, Ethereum y Monero. Cualquier persona, sin necesidad de permisos de acceso puede formar parte de cualquiera de ellas, tanto para realizar transacciones de carácter privado como la compra de bienes y servicios que admitan como moneda de pago bitcoins, ethers o moneros o también para el “minado” de bloques con el que conseguir un rédito económico.



Ilustración 3: Logotipos de Bitcoin, Ether y Monero

2.1.2 Blockchain privadas

Tanto este modelo como el de consorcio se podrían considerar como la segunda generación de Blockchain. El modelo privado consiste en iniciativas dirigidas por una organización, en donde el acceso a la red si está controlado por una entidad, y las identidades de los participantes si son conocidas. En la actualidad el entorno más extendido para la generación de una Blockchain de carácter privado sería el Hyperledger⁶. Este proyecto está liderado por la Fundación Linux, de la cual son miembros compañías como IBM[®], SAP[®], Oracle[®], Google[®] o Intel[®], las cuales aportan al mismo su apoyo económico y técnico.

En este entorno las compañías que lo deseen pueden desarrollar su proyecto basado en una blockchain privada con sus reglas de negocio y objetivos propios, sin la necesidad de realizar de manera obligatoria transacciones con criptomonedas.

2.1.3 Blockchain de consorcio

Este modelo se puede definir como una combinación de los dos anteriores, pero con la característica de que está dirigido a sectores económicos específicos o a grupos de compañías con intereses comunes.

Los ejemplos que se pueden dar serían:

- Consorcio B3i⁷: Esta es una iniciativa creada por un consorcio de empresas de seguros, y que tiene como objetivo la creación de estándares y protocolos entre ellas con el objeto de disminuir los costes operativos derivados de los necesarios intercambios de información entre ellas.
- Alastria: La definición que indica en su web (<https://alastria.io>) nos da una idea muy clara de su objetivo: “Alastria es una asociación sin ánimo de lucro que fomenta la Economía Digital mediante el desarrollo de Blockchain.” (Alastria-Presentación corporativa.pdf, p. 1).

⁶ <https://www.hyperledger.org/>

⁷ <https://b3i.tech/home.html>

Este es un proyecto abierto a cualquier empresa, independientemente de su tamaño, que desee desarrollar proyectos dentro de un modelo de economía digital descentralizada. Para participar en el proyecto es necesario ser miembro del consorcio, y en la actualidad sólo está abierto a personas jurídicas.

2.2 Características

Esta “cadena de bloques” posee las siguientes características:

- **Descentralizada** (*Trustless*): Todos los nodos son responsables de la gestión y modificación de la información al mismo tiempo. Esto significa que cualquier servicio o aplicación seguirá funcionando si un nodo o grupo de nodos deja de estar disponible, en este caso la información seguirá fluyendo al persistir en el resto de nodos de la red.
- **Distribuida** (*Many to many*): Cualquier nodo perteneciente a la blockchain está conectado al resto de nodos de la red. Con la aplicación estricta de este concepto, en una Blockchain se hace casi imposible manipular los datos almacenados en ella, dado que todos los nodos tienen una copia de los mismos y los cambios no permitidos podrían ser rechazados al ser dados como *no válidos*.
- **Consistencia de datos** (*Proof of work*): Tanto en **Bitcoin** como en **Ethereum** los nodos pertenecientes a la red verifican la validez de cada transacción realizada en ella, a través de un **algoritmo de consenso**. En el caso de estas dos redes, este algoritmo se denomina **Proof of Work**, aunque en la actualidad la red Ethereum está comenzando a migrar al algoritmo **Proof of Stake**. En capítulos posteriores profundizaremos en estos dos conceptos.
- **Inmutable** (*Ledger*): Los datos almacenados en la red no pueden ser eliminados. Podrán ser modificados en posteriores operaciones, pero siempre existirá un registro histórico con todos los datos a través de la cadena de bloques.

2.2.1 ¿Qué es un bloque?

Según se define en https://www.sinergiablockchain.org/_/Doc/FAQ.pdf:

“Un bloque es un conjunto de transacciones confirmadas e información adicional que se ha incluido en la cadena de bloques. Cada bloque que forma parte de la cadena (excepto el bloque generatriz, que inicia la cadena) está formado por:

- *Un código alfanumérico que enlaza con el bloque anterior.*
- *El “paquete” de transacciones que se incluye.*
- *Otro código alfanumérico que enlazará con el siguiente bloque.”*

La imagen siguiente ilustración muestra el concepto de una “cadena de bloques”.

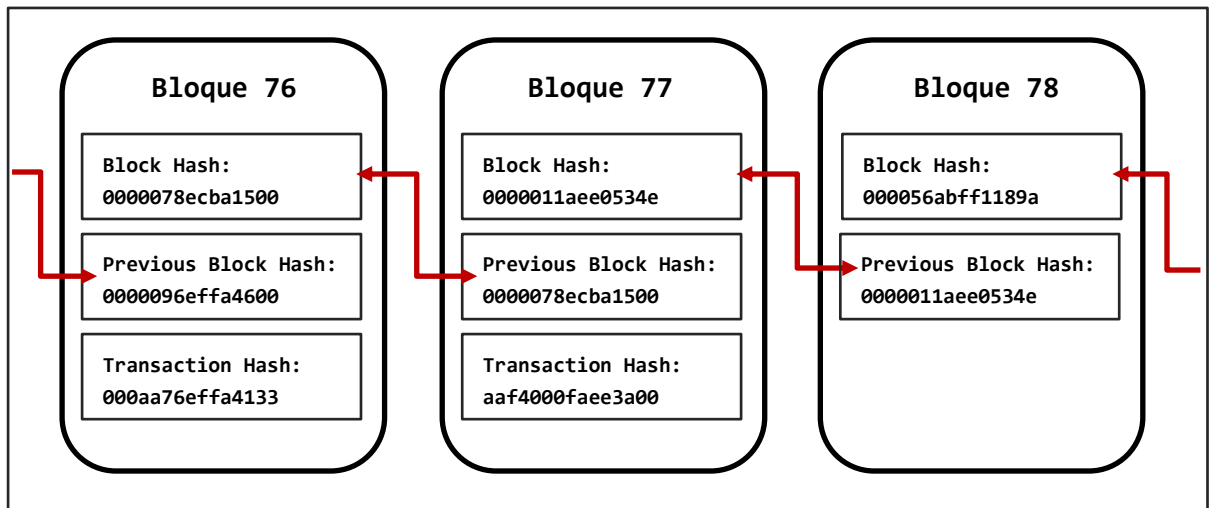


Ilustración 4: Representación de la cadena de bloques.

Pero como en todo siempre hay un principio, y la blockchain no es una excepción, existe un bloque generatriz denominado **Génesis**, a partir del cual se encadenan el resto.

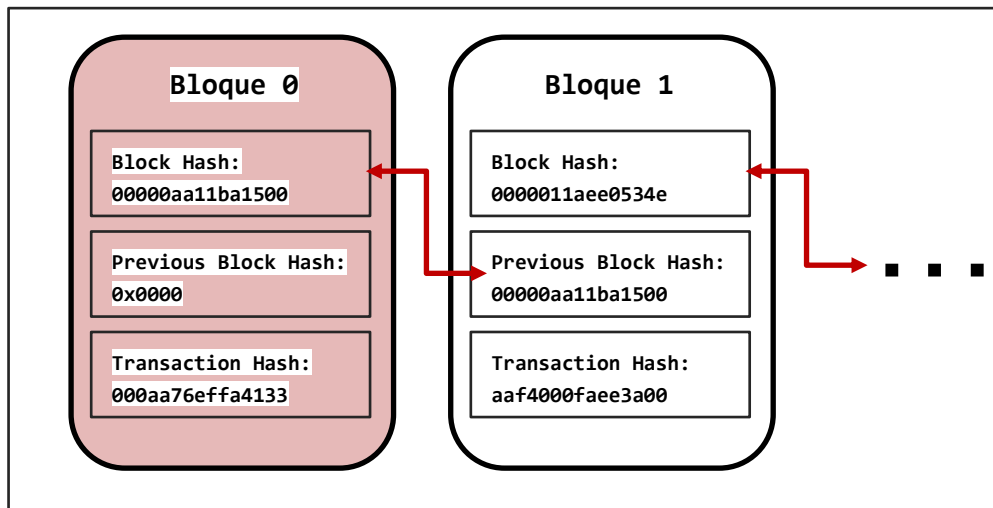
```

"config": {
  "chainId": 0,
  "homesteadBlock": 0,
  "eip155Block": 0,
  "eip158Block": 0
},
"alloc" : {},
"coinbase" : "0x0000000000000000000000000000000000000000",
"difficulty" : "0x20000",
"extraData" : "",
"gasLimit" : "0x2fef8",
"nonce" : "0x0000000000000042",
"mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
"parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000",
"timestamp" : "0x00"
}

```

Ilustración 5: Ejemplo de bloque genesis.json

Este bloque se forma en el momento de la creación de la Blockchain, y su estructura está definida en un fichero en formato JSON que recibe el nombre de *genesis.json*.

*Ilustración 6: Bloque Genesis*

Como se observa en las imágenes anteriores, los identificadores de los bloques son códigos generados por funciones resumen o *funciones hash*. Estos identificadores son los elementos que sirven de enlaces entre los bloques. En cada uno de ellos hay un “Previous Block hash” que permiten enlazar un bloque con el anterior. Además se almacena el “Transaction hash” que contiene la transacción realizada y validada.

Los bloques se sitúan en los NODOS, y estos no son más que cada una de las máquinas que componen la red, y en donde existe una copia de todas las transacciones realizadas hasta el momento. Cada bloque se añade a la blockchain de manera cronológica y con una marca de tiempo o *timestamp*. Cada vez que una transacción es validada, esta se almacena de manera permanente y sin posibilidad de ser modificada o eliminada.

Ahora bien, ¿quién o que valida cada transacción? Cuando a la blockchain le llega una solicitud para realizar una transacción, realiza un proceso de *consenso*, éste es un acuerdo entre los nodos participantes en la red, en el que se decide que la información que se ha almacenado es cierta, y por lo tanto puede pasar a formar parte de la cadena de bloques. Estos acuerdos entre bloques se alcanzan a través de los *algoritmos de consenso*, éstos obligan a que el protocolo de la blockchain sea respetado, y por consiguiente dan fiabilidad a cualquier transacción que se realice.

Los encargados de ejecutar estos algoritmos, son los nodos mineros o *miners*, y ellos gastan energía y tiempo de cómputo para validar la transacción.

¿Qué algoritmos de consenso existen? La respuesta es que bastantes, pero centrándonos en la red Bitcoin y Ethereum, éstas dos emplean los denominados *Prueba de trabajo* (Proof of work) y *Prueba de participación* (Proof of stake). En la actualidad *Proof of work* es el algoritmo más utilizado, pero la red Ethereum está migrando a *Proof of stake* debido al alto consumo de energía al que obliga el primero.

2.2.2 Proof of work

Fue el primer algoritmo de consenso creado para la red Bitcoin, y aquí los mineros realizan la labor de validación resolviendo un problema matemático. Es en este punto donde entra la potencia de cómputo de cada minero participante. Se establece una competición para lograr resolver el problema antes que ninguno, pues aquel que lo logre será el que consiga la recompensa en forma de criptomoneda, *bitcoins* para la red Bitcoin o *ethers* para la red Ethereum.

Como se ha dicho anteriormente, este algoritmo trata de resolver un problema matemático, que permita la validación del bloque. El proceso que sigue este algoritmo es el siguiente:

1. Existe un área de espera denominada *mempool*, en donde se alojan todas las transacciones pendientes de ser validadas por los mineros. Cada transacción lleva asociado un hash y un *fee* o tasa, que el creador de la transacción debe pagar para que ésta sea reflejada en la cadena de bloques y que será cobrada por el minero que resuelva el problema.



Ilustración 7: *mempool*

2. Cada minero agrupa transacciones, según un criterio propio, dentro de lo que se denomina un bloque candidato.
3. A cada bloque candidato se le asigna una dificultad, en nuestro caso podemos suponer que es cuatro. Esto significará que el cifrado final correspondiente al bloque debe de contener al menos cuatro ceros al inicio de su secuencia.
4. Para realizar la labor de minado es necesario utilizar un valor aleatorio, que se usa una sola vez, denominado *nonce* y que se añade al final del bloque. Con este valor y el algoritmo SHA256, se cifra el bloque y se halla un valor alfanumérico hexadecimal, por ejemplo:

454aa1487cc8b3137859531f98e999289269490579d495d5cb17680145ce0d52.

5. Como la dificultad del bloque está establecida en cuatro, si el valor obtenido no tiene cuatro ceros por delante, se considera inválido, por lo que es necesario emplear otro nonce y realizar de nuevo el cifrado.
6. Es decir, hay que realizar este proceso hasta encontrar un valor válido, por ejemplo: 0000e011db58748cfb2e44b007ec91bb1a0f0073dab58e5ddb4c013a8b5932cc.
7. Entonces, el objetivo del minero será el de encontrar el valor del nonce que proporcionará en hash correcto antes que el resto, y con ello llevarse la recompensa y las tasas pagadas por el emisor de cada transacción. Como es obvio, la dificultad de encontrar un valor que satisfaga esta condición es muy alta, con lo que en este punto es donde entra la potencia de cómputo de cada minero, para encontrar la solución en primera posición.
8. El bloque hallado será añadido a la blockchain y el proceso se reiniciará por parte de todos los mineros a partir de las transacciones almacenadas en la mempool.

2.2.3 Proof of stake

Este es el otro algoritmo de consenso, que a diferencia del proof of work, lo que pretende es premiar la participación en la red, en lugar de la potencia de cálculo. Es de reciente creación, de hecho surgió en el año 2011 y su primera implementación fue para la criptomoneda Peercoin en el año 2012.

En este sistema, la probabilidad de ser elegido para validar un bloque, y por lo tanto para recibir la recompensa, es un valor aleatorio pero que en cierta medida viene determinado por el porcentaje de monedas que cada minero posee dentro de la red, por lo tanto si un minero tiene el 0,05% de monedas, tendrá 0,05% posibilidades de ser elegido para validar el bloque. La prueba de validación pretende evitar el alto consumo de energía de PoW, así como la centralización de recursos de hardware por parte de un número reducido de mineros.

2.3 IPFS

La definición más ortodoxa que podemos encontrar sobre este sistema de almacenamiento es la que se ofrece en el Libro Blanco de la web <https://ipfs.io/>:

“The Inter-Planetary File System (IPFS) is a peer-to-peer distributed file system, that seeks to connect all computing devices with the same system of files.” (Juan Benet, 2014, p.1).

Se define como un protocolo en el que cada fichero almacenado se identifica con un hash único, denominado *hash criptográfico*, esto impide la duplicación de los ficheros en la red. Cuando se busca un fichero lo que se está localizando es el nodo que se identifica detrás de este hash.

2.4 DApp. ¿Qué es?

DApp es el acrónimo en inglés de Decentralized Application, y la definición más consensuada que podemos encontrar es la de una aplicación que se ejecuta en una red peer-to-peer (P2P) y que aprovecha la capacidad de los Smartcontracts para realizar transacciones de carácter inmutable y confiable.

Esta capacidad de aprovechar las funcionalidades de los “contratos inteligentes” es la que ha permitido su extensión en el ámbito de la red Ethereum, pues en ésta Blockchain es donde han alcanzado su máxima implantación.

Poseen una serie de características de carácter general que se enumeran a continuación:

- Son proyectos de código abierto.
- Al ser aplicaciones descentralizadas, no hay un gestor al cargo.
- Utilizan la estructura física y lógica de una Blockchain pública o privada para poder funcionar.
- Todas las transacciones son inmutables y no es posible su falsificación.
- Son un formato de muy reciente generación, con lo que su estructura y formato están en constante evolución.
- No hay un límite de usuarios.

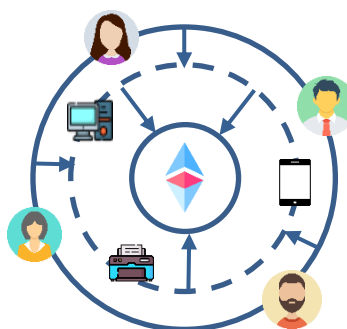


Ilustración 8: Esquema figurado DApp

2.5 Smartcontracts

Un Smartcontract es un código informático, que se ejecuta en la Blockchain y cuyas transacciones y cuya lógica definida es la que sirve de soporte como “lógica de negocio” a la aplicación que se apoya en él.

Una de las características fundamentales es que, como se ha descrito anteriormente, las transacciones realizadas por ellos son inmutables y por lo tanto confiables debido al entorno en donde se ejecutan.

Los Smartcontracts que se ejecutan en Petchain están desarrollados en lenguaje Solidity, el cual comparte muchas similitudes con Javascript en codificación y tipado de datos.

3 Petchain. Especificación de requisitos.

Petchain es una DApp destinada al sector veterinario que permite la gestión de las identidades de mascotas domésticas, así como la de sus propietarios. La aplicación se basa en un desarrollo web tradicional, pero con la característica fundamental de estar implementada en una Blockchain bajo una red Ethereum y almacenado las identidades de las mascotas bajo ficheros JSON que son enviados al sistema IPFS.

Petchain gestiona de las identidades de las mascotas domésticas dando de alta, modificando los datos y gestionando la baja de los datos de los animales, bajo la ejecución de Smartcontracts creados en lenguaje Solidity.

3.1 Propósito del documento

El presente documento pretende realizar una descripción de los requisitos informativos, funcionales y no funcionales de Petchain, con el fin de comprender y describir su funcionamiento, así como la enumeración de los participantes en el desarrollo, de cada una de sus partes y conseguir al final del mismo, una relación entre las partes y los creadores de las mismas.

Pretende ser una descripción de los requisitos informativos, funcionales, no funcionales del proyecto, con el objetivo de ofrecer a los lectores del documento del TFG una visión clara de la aplicación y las partes que lo componen.

3.2 Alcance del documento

El presente documento tiene como alcance especificar:

- La declaración de los participantes en el proyecto.
- Definir de los objetivos de la aplicación.
- Proveer una descripción funcional del sistema que permita la mejor comprensión del mismo al equipo de evaluadores del proyecto TFG.
- Especificar de los requisitos informativos, funcionales y no funcionales que deberá cumplir la aplicación
- Establecer los límites de funcionales de Petchain.

3.3 Participantes en el proyecto

3.3.1 Presentación de participantes

| | |
|---------------------|---|
| Participante | Miriam Martínez Muñoz |
| Organización | Universidad Internacional de la Rioja |

| | |
|-------------------------|---|
| Rol | Dirección de proyecto |
| Es desarrollador | Sí |
| Es cliente | No |
| Es usuario | No |
| Comentarios | Directora del proyecto de fin de grado. |

Tabla 2: Dirección del proyecto

| | |
|-------------------------|---|
| Participante | Carlos Escario Bajo |
| Organización | Universidad Internacional de la Rioja |
| Rol | Analista-Desarrollador |
| Es desarrollador | Sí |
| Es cliente | No |
| Es usuario | No |
| Comentarios | Ninguno |

Tabla 3: Analista y desarrollador del proyecto.

| | |
|-------------------------|---|
| Participante | Pedro Escario Bajo |
| Organización | Veterinaria García-Ochoa |
| Rol | Usuario |
| Es desarrollador | No |
| Es cliente | No |
| Es usuario | Sí |
| Comentarios | Trabajador en la Veterinaria García-Ochoa |

Tabla 4: Colaborador del proyecto.

| | |
|-------------------------|--|
| Participante | Iciar García-Ochoa |
| Organización | Veterinaria García-Ochoa |
| Rol | Usuario |
| Es desarrollador | No |
| Es cliente | No |
| Es usuario | Sí |
| Comentarios | Propietaria de la Veterinaria García-Ochoa |

Tabla 5: Dirección facultativa y Veterinaria

3.4 Organizaciones involucradas

| | |
|---------------------|--|
| Organización | Universidad Internacional de la Rioja |
| Dirección | Av. de la Paz, 137, 26006 Logroño, La Rioja |
| Teléfono | N/A |
| Fax | N/A |
| Comentarios | www.unir.net |

Tabla 6: Organización 1.

| | |
|---------------------|--|
| Organización | Veterinaria García-Ochoa |
| Dirección | Carretera de Casillas 6 local 3, 05420 Sotillo de la Adrada, Ávila |
| Teléfono | N/A |
| Fax | N/A |
| Comentarios | Ninguno |

Tabla 7: Organización 2

3.5 Descripción del sistema

3.5.1 Perspectiva del producto

Petchain tiene como objeto mostrar las capacidades que posee la Blockchain para el desarrollo de aplicaciones de gestión tradicionales. No es objeto de la misma ofrecer la visión de que el modelo que se presenta "es mejor" que cualquier desarrollo web tradicional, sino que como nuevo paradigma de desarrollo, es necesario tenerlo en cuenta como un futuro modelo para la generación de aplicaciones de gestión corporativas.

Su ejecución se realiza íntegramente en un entorno local, permitiendo este modelo, la visualización de cada una de las partes que componen el desarrollo. Esto es posible gracias al uso del framework Truffle, que con el servidor Ganache que lleva incorporado, ofrece la posibilidad de crear una red Ethereum y con ello poder desarrollar, desplegar y testear Smartcontracts desarrollados con lenguaje Solidity. Por último, el almacenamiento de los objetos JSON generados se realiza bajo el protocolo IPFS.

3.5.2 Funciones del producto

Petchain es una aplicación que debe de permitir a los veterinarios:

- Identificar a las mascotas domesticas mediante un código alfanumérico único, así como asociarlas a una persona que actuará como dueño, y que estará identificado mediante DNI o Pasaporte y los datos personales (nombre, apellidos, domicilio, y teléfono de contacto).

- Modificar los datos sanitarios de vacunaciones,
- Modificar los datos personales del dueño de la mascota.
- Dar de baja a las mascotas del sistema.
- Mostrar un listado de mascotas.
- Visualizar el saldo en ethers (ETH) y weis de cada uno de los veterinarios correspondientes a las transacciones realizadas.

3.5.3 Características del usuario

Los usuarios de Petchain serán los veterinarios, que a nivel nacional, deben de realizar acciones de tipo administrativo a la hora de gestionar las identificaciones de animales de compañía.

Como perfil tecnológico, deben tener competencias a la hora de manejar formularios web, sin necesitar unos conocimientos técnicos profundos, pues el modelo visual no difiere de cualquier página web existente en el mercado.

Por otro lado el tipo de información que es necesaria cumplimentar en la web, está dentro del dominio del sector veterinario, con lo que su cumplimentación no debería presentar problemas para el colectivo que usará la aplicación.

3.5.4 Restricciones

La visualización de Petchain debe ser óptima en toda clase de dispositivos, por lo tanto la estructura de las pantallas se debe de adaptar a la resolución de la pantalla en donde se ejecute, siendo el rango de estas desde dispositivos Smartphone, Tablets y PCs.

A nivel de seguridad, es necesario implementar características que permitan la autenticación y la autorización de los usuarios que intenten acceder a la aplicación, pero al ser una ejecución en un entorno local no es obligatorio el diseño de una base de datos SQL o NoSQL, que albergue sus credenciales de acceso.

3.6 Descripción técnica del sistema

3.6.1 Front-End

La web se desarrollará bajo el modelo de visualización responsive, por lo que su diseño se deberá adaptar al tipo de resolución de la pantalla desde la que se esté accediendo. Para conseguir este objetivo se empleará en su diseño el framework open source [Twitter Bootstrap](#), ver. 4.3.1.

El desarrollo de las interfaces de usuario se realizará con [HTML5](#) estándar, [CSS3](#) y la programación de los scripts de cliente se realizará con JavaScript ([ECMA Script 6](#)) y el

framework de cliente [jQuery](#) versión 3.1.1., siendo uno de los objetivos del desarrollo el de aprovechar al máximo el paradigma de programación [MVC](#) (Model View Controller).

3.6.2 Seguridad

Como antes se indicaba, es necesario implementar un sistema que verifique la autenticación de los usuarios al sistema, así como la autorización para la realización de acciones asociadas al rol que tenga determinado. El sistema de credenciales empleado para el proyecto actual debe quedar preparado para poder implementar una conexión a una base de datos [NoSQL](#) en el caso de que Petchain se instale en un entorno productivo. En el modelo de ejecución actual, sólo se exigirá que los usuarios estén definidos en un fichero [JSON](#) almacenado dentro de la estructura local de proyecto.

3.6.3 Back-End

La aplicación se ejecutará en una Blockchain desplegada en un entorno local. La red estará funcionando desplegada en un servidor Ganache, el cual es un componente del framework de desarrollo y ejecución Truffle.

La conexión entre la capa de Back-End y el Front-End es posible mediante el uso del framework Web3.js. Este entorno, desarrollado íntegramente en Javascript, permite la interacción de Smartcontracts con el JavaScript alojado en el cliente y la ejecución de la lógica en ellos definida.

Por último la interacción de ambas capas permitirá la creación y modificación de uno o varios ficheros JSON, los cuales harán como soporte No SQL, el cual se implementará como un fichero físico en IPFS.

3.6.4 Entorno de ejecución

La aplicación debe de ejecutarse en cualquier navegador estándar (IE11, Chrome o Firefox) en un servidor local o localhost.

Para la ejecución de funcionalidades como la seguridad, lectura y escritura de ficheros y paso de mensajes entre pantallas, es necesario la realización de solicitudes POST y GET, por lo la aplicación se va a desplegar con un servidor ExpressJS y bajo un entorno de ejecución Node.js.

3.7 Objetivos del sistema

| | |
|---------------------|---|
| OBJ-0001 | Alta de mascotas |
| Versión | 1.0 (04/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Descripción | El sistema deberá permitir el alta de una mascota en el sistema, asociándola a un código de chip. |
| Subobjetivos | Ninguno |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | PD |
| Comentarios | Ninguno |

Tabla 8: Objetivo de Alta de mascotas

| | |
|---------------------|--|
| OBJ-0002 | Alta de dueños de mascotas |
| Versión | 1.0 (04/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo |
| Descripción | El sistema deberá permitir el dar de alta a una persona, asociándola a una mascota. |
| Subobjetivos | Ninguno |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 9: Objetivo Alta de dueños.

| | |
|-----------------|--|
| OBJ-0003 | Modificación de datos |
| Versión | 1.0 (04/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo |

| | |
|---------------------|---|
| Descripción | El sistema deberá permitir modificar los datos asociados a las mascotas y a sus dueños. |
| Subobjetivos | Ninguno |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |

Tabla 10: Objetivo de Modificación de datos

| | |
|---------------------|---|
| OBJ-0004 | Baja de mascotas |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Descripción | El sistema deberá permitir dar de baja a una mascota del sistema, por motivos de defunción. |
| Subobjetivos | Ninguno |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 11: Objetivo de Baja de mascotas

| | |
|---------------------|---|
| OBJ-0005 | Consulta de datos de la mascota y su propietario. |
| Versión | 1.0 (04/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Descripción | El sistema deberá permitir la consulta de los datos de la mascota y de su propietario. |
| Subobjetivos | Ninguno |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 12: Objetivo de Consulta de datos de una mascota

| | |
|---------------------|--|
| OBJ-0006 | Consulta de saldo |
| Versión | 1.0 (04/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | El sistema deberá consultar el saldo en ethers o weis que cada veterinario participante en el sistema, tiene por realizar acciones dentro del mismo. |
| Subobjetivos | Ninguno |
| Importancia | Importante |
| Urgencia | Puede esperar |
| Estado | Validado |
| Estabilidad | Media |
| Comentarios | Ninguno |

Tabla 13: Objetivo de Consulta de saldo

| | |
|---------------------|---|
| OBJ-0007 | Control de acceso |
| Versión | 1.0 (06/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | Sólo se deberá permitir el acceso a los usuarios que estén registrados en el sistema |
| Subobjetivos | Ninguno |
| Importancia | Vital |
| Urgencia | Hay presión |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 14: Objetivo de Control de acceso.

| | |
|--------------------|---|
| OBJ-0008 | Visualización en diferentes dispositivos |
| Versión | 1.0 (06/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | Que exista una correcta visualización para trabajar en dispositivos PC, Tablet y Smartphone |

| | |
|---------------------|----------------|
| Subobjetivos | Ninguno |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 15: Objetivos de visualización

| | |
|---------------------|--|
| OBJ-0009 | Navegación por el sistema |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | Que se permita poder navegar a otra opción independientemente del punto en donde nos encontremos de la aplicación. |
| Subobjetivos | Ninguno |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 16: Objetivo de la navegación de la aplicación

| | |
|---------------------|---|
| OBJ-00010 | Traspasar ETH |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | Que se produzca un intercambio de ETH cuando existan modificaciones en los datos de las mascotas. |
| Subobjetivos | Ninguno |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 17: Objetivo de traspaso de ETH

3.8 Catálogo de requisitos del sistema.

3.8.1 Requisitos de información

Los requisitos de información son formas especializadas de requisitos que van a permitir la introducción de datos relativos a los actores que van a participar o componer el sistema.

| | | |
|--------------------------------|--|---------------|
| IRQ-0001 | Datos de usuario veterinario | |
| Versión | 1.0 (05/05/2019) | |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo | |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa | |
| Dependencias | <ul style="list-style-type: none"> • [FRQ-0001] Acceso al sistema | |
| Descripción | El sistema deberá almacenar la información correspondiente a los usuarios veterinarios que usen el sistema. En concreto: | |
| Datos específicos | <ul style="list-style-type: none"> • Nombre de usuario • Contraseña del usuario • Rol • Número de colegiado • Colegio | |
| Tiempo de vida | Medio | Máximo |
| | Años | Años |
| Ocurrencias simultáneas | Medio | Máximo |
| | Años | Años |
| Importancia | Vital | |
| Urgencia | Inmediatamente | |
| Estado | Finalizado | |
| Estabilidad | Alta | |
| Comentarios | Ninguno | |

Tabla 18: Requisito de Información del veterinario

| | | |
|---------------------|--|--|
| IRQ-0002 | Datos de mascotas | |
| Versión | 1.0 (05/05/2019) | |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo | |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo | |
| Dependencias | Ninguno | |
| Descripción | El sistema deberá almacenar la información correspondiente a las mascotas que se añaden al sistema. En concreto: | |

| | | |
|--------------------------------|---|---------------|
| Datos específicos | <ul style="list-style-type: none"> • N° de identificación • Fecha de implantación • Nombre • Raza Peligrosa • Fecha de Nacimiento • Capa • Pelo • Sexo • Pasaporte animal • Aptitud • Última vacunación antirrábica • Fecha de la última revisión • Número de colegiado última revisión • Cicatrices • Numero de certificado | |
| Tiempo de vida | Medio | Máximo |
| | Años | Años |
| Ocurrencias simultáneas | Medio | Máximo |
| | Años | Años |
| Importancia | Importante | |
| Urgencia | Inmediatamente | |
| Estado | Validado | |
| Estabilidad | Estable | |
| Comentarios | Ninguno | |

Tabla 19: Requisitos de información de mascotas

| | |
|--------------------------|---|
| IRQ-0003 | Datos del dueño |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá almacenar la información correspondiente al dueño de la mascota. En concreto: |
| Datos específicos | <ul style="list-style-type: none"> • Nombre • Apellidos • Número de documento • Teléfono • Provincia • Municipio • Código Postal • País |

| | | |
|--------------------------------|----------------|---------------|
| Tiempo de vida | Medio | Máximo |
| | Años | Años |
| Ocurrencias simultáneas | Medio | Máximo |
| | Años | Años |
| Importancia | Importante | |
| Urgencia | Inmediatamente | |
| Estado | Validado | |
| Estabilidad | Estable | |
| Comentarios | Ninguno | |

Tabla 20: Requisitos de información de propietarios.

| | | |
|--------------------------------|--|---------------|
| IRQ-0004 | Saldo | |
| Versión | 1.0 (05/05/2019) | |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo | |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo | |
| Dependencias | Ninguno | |
| Descripción | El sistema deberá almacenar la información correspondiente a al saldo en ethers o weis correspondiente a los usuarios veterinarios que usen el sistema. En concreto: | |
| Datos específicos | <ul style="list-style-type: none"> • Cuenta Ethereum • Saldo en weis • Número de colegiado | |
| Tiempo de vida | Medio | Máximo |
| | Años | Años |
| Ocurrencias simultáneas | Medio | Máximo |
| | Años | Años |
| Importancia | Importante | |
| Urgencia | Inmediatamente | |
| Estado | Validado | |
| Estabilidad | Estable | |
| Comentarios | Ninguno | |

Tabla 21: Requisitos de información del saldo en ETH

3.8.2 Requisitos funcionales

Ian Sommerville en su libro “Ingeniería de Software 9” define los requisitos funcionales como: “servicios que el sistema debe proveer, de cómo debería reaccionar el sistema a entradas particulares y de cómo debería comportarse el sistema en situaciones específicas.

En algunos casos, los requerimientos funcionales también explican lo que no debe hacer el sistema” (Pearson Education, 2011, págs. 84 - 85).

Estos requisitos nos darán una visión concreta de cuáles son los objetivos de funcionamiento de Petchain, y que además tienen su correspondencia con los Casos de uso descritos en la sección 5.10.

| | |
|---------------------|---|
| FRQ-0001 | Acceso al sistema |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá permitir el acceso a los usuarios a través de un formulario en el que se debe introducir un nombre de usuario y una contraseña. Si los datos introducidos no son correctos, el sistema deberá mostrar en la pantalla un mensaje de error. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 22: Requisito funcional de acceso al sistema

| | |
|---------------------|---|
| FRQ-0002 | Registro en la Blockchain |
| Versión | 1.0 (01/06/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá mostrar en la pantalla principal de la aplicación la primera vez que un veterinario accede a la aplicación, una ventana solicitándole su registro en la Blockchain, mediante la introducción de su dirección de correo electrónico. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 23: Requisito funcional de registro en la Blockchain

| | |
|---------------------|---|
| FRQ-0003 | Identificación de veterinarios |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Dependencias | Ninguno |
| Descripción | El sistema deberá mostrar el nombre y el colegio en donde está inscrito del Veterinario que ha accedido a la aplicación, en las pantallas de alta, modificación y baja de las mascotas. |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 24: Requisito funcional de identificación de veterinarios.

| | |
|--------------------|--|
| FRQ-0004 | Alta de una mascota |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Descripción | <p>El sistema deberá Como veterinario quisiera dar de alta una mascota en el sistema, mediante un formulario en la pantalla de "Alta de Mascotas", que contenga los campos imprescindibles para identificar a un animal, en concreto sería necesarios:</p> <ul style="list-style-type: none"> • Nombre del animal. • Fecha de implantación del chip. • Alta en la base de datos. • Número de identificación. • Raza. • Peligrosidad. • Fecha de nacimiento. • Capa. • Pelo. • Nº de pasaporte. • Aptitud. |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 25: Requisito funcional de alta de una mascota

| | |
|---------------------|--|
| FRQ-0005 | Alta de datos de un propietario |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Dependencias | Ninguno |
| Descripción | <p>El sistema deberá permitir al veterinario dar de alta al dueño de una mascota con los datos imprescindibles para identificar al dueño de un animal, en concreto sería necesarios:</p> <ul style="list-style-type: none"> • Nombre. • Apellidos. • Dirección: Calle, Ciudad, CP y País. • Id del documento de identificación. • Teléfono. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 26: Requisito funcional de alta de un propietario

| | |
|---------------------|--|
| FRQ-0006 | Modificación de datos de una mascota |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Dependencias | Ninguno |
| Descripción | <p>Como veterinario quiero poder modificar los datos sanitarios de una mascota, en concreto:</p> <ul style="list-style-type: none"> • Nº de pasaporte. • Fecha de revisión. • Estado. • Cicatrices. • Vacunas. • Observaciones |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 27: Requisito funcional de modificación de datos de una mascota

| | |
|---------------------|--|
| FRQ-0007 | Modificación de datos de un propietario |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa |
| Dependencias | Ninguno |
| Descripción | <p>Como veterinario quisiera poder modificar los datos del dueño de una mascota, cambiando los datos imprescindibles que permitan identificar al dueño de un animal, en concreto sería necesarios:</p> <ul style="list-style-type: none"> • Nombre. • Apellidos. • Dirección: Calle, Ciudad, CP y País. • Id del documento de identificación. • Teléfono. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 28: Requisito funcional para la baja de una mascota

| | |
|---------------------|---|
| FRQ-0008 | Consultar ETH traspasados |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | <p>El sistema deberá mostrar un listado con las transacciones recibidas de otros veterinarios por haber modificado los datos de una mascota dada de alta por él y el saldo total en ETH de su cuenta. ,</p> |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 29: Requisito funcional para la visualización de weis/ETH traspasado

| | |
|-----------------|------------------------|
| FRQ-0009 | Traspaso de ETH |
| Versión | 1.0 (05/05/2019) |

| | |
|---------------------|---|
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá traspasar ETH cuando se haga una modificación de los datos de una mascota, desde el veterinario modificador hasta el veterinario que dio de alta a la mascota, que estarán almacenados en el balance del SmartContract. |
| Importancia | Importante |
| Urgencia | Puede esperar |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 30: Requisito funcional Traspaso de ETH o weis.

| | |
|---------------------|--|
| FRQ-0011 | Consultar datos |
| Versión | 1.0 (01/06/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Iciar García-Ochoa • Pedro Escario Bajo |
| Dependencias | Ninguno |
| Descripción | Como veterinario, me gustaría que el sistema permitiese consultar los datos de una mascota y su propietario y que además no puedan ser modificados |
| Importancia | Importante |
| Urgencia | Puede esperar |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 31: Requisito funcional para consultar datos

3.8.3 Requisitos no funcionales

La descripción más exacta que podemos obtener de los requisitos no funcionales es la expresada en el libro Ingeniería de Software, Ian Sommerville (2009):

Son requerimientos que no se relacionan directamente con los servicios específicos que el sistema entrega a sus usuarios. Pueden relacionarse con propiedades emergentes del sistema, como fiabilidad, tiempo de respuesta y uso de almacenamiento. De forma alternativa, pueden definir restricciones sobre la implementación del sistema, como las capacidades de los dispositivos I/O o las representaciones de datos usados en las interfaces con otros sistemas (pág. 87).

| | |
|---------------------|---|
| NFR-0001 | Visualización |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá ajustar la visualización de las interfaces gráficas en función del modelo de desarrollo responsive. Es decir será el cliente quien detecte el tipo de dispositivo desde el que se está accediendo a la web. |
| Importancia | Quedaría bien |
| Urgencia | Importante |
| Estado | Finalizado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 32: Requisito no funcional de visualización de Petchain.

| | |
|---------------------|---|
| NFR-0002 | Moneda de pago |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá permitir el pago en weis, siendo la conversión en ethers de $1 \text{ ether} = 1 \text{ e}^{-18} \text{ wei}$. |
| Importancia | Importante |
| Urgencia | Por definir |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 33: Requisito no funcional del pago en weis

| | |
|---------------------|--|
| NFR-0003 | Tiempo de sesión |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá controlar que el tiempo que puede estar una sesión inactiva, será como máximo de 15 minutos. |

| | |
|--------------------|---------------|
| Importancia | Importante |
| Urgencia | Puede esperar |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 34: Requisito no funcional del tiempo de sesión.

| | |
|---------------------|--|
| NFR-0004 | Usuarios autorizados |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá tener almacenados los usuarios de la aplicación en un fichero en formato JSON , en donde se indique su número de colegiado, nombre de usuario, contraseña y número de colegiado. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Pendiente de validación |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 35: Requisito no funcional de usuarios autorizados

| | |
|---------------------|--|
| NFR-0005 | Número de cuentas |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá tener definidas por defecto al menos 100 cuentas en el servidor Ganaché. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 36: Requisito no funcional del nº de cuentas en Ganache

| | |
|---------------------|--|
| NFR-0006 | Campos obligatorios |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá mostrar en pantalla un mensaje que informando de aquellos campos que son obligatorios y no han sido rellenados en el momento de enviar el formulario y señalándolos en color rojo. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Finalizado |
| Estabilidad | Por definir |
| Comentarios | Ninguno |

Tabla 37: Requisito no funcional de campos obligatorios

| | |
|---------------------|---|
| NFR-0007 | Opciones de navegación |
| Versión | 1.0 (01/06/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El sistema deberá mostrar en todo momento las opciones de navegación por la aplicación o un punto de acceso siempre visible que permita el acceso a las mismas. |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 38: Requisito no funcional de definición de opciones de navegación

| | |
|---------------------|---|
| NFR-0008 | Carga de datos estáticos |
| Versión | 1.0 (06/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |

| | |
|--------------------|--|
| Descripción | El sistema deberá cargar los datos estáticos a partir de ficheros JSON almacenados en el sistema |
| Importancia | Importante |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 39: Requisito no funcional de carga de datos estáticos.

| | |
|---------------------|--|
| NFR-0009 | Validación de código |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | El código desarrollado para la aplicación no deberá contener errores de codificación, por lo que todos los ficheros estáticos html y css deberán superar las pruebas de validación que establece el consorcio W3C en https://validator.w3.org |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 40: Requisito no funcional de obligación de validar el código.

| | |
|---------------------|---|
| NFR-0010 | Confirmación de transacciones |
| Versión | 1.0 (05/05/2019) |
| Autores | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Fuentes | <ul style="list-style-type: none"> • Carlos Escario Bajo |
| Dependencias | Ninguno |
| Descripción | Todas las acciones realizadas se deben ser confirmadas o aceptadas por los usuarios en ventanas con mensajes de advertencia |
| Importancia | Vital |
| Urgencia | Inmediatamente |
| Estado | Validado |
| Estabilidad | Alta |
| Comentarios | Ninguno |

Tabla 41: Requisito no funcional de confirmación de transacciones

3.9 Matriz de rastreabilidad

3.9.1 Matriz de rastreabilidad

| TRM-0001 | | Requisitos Funcionales (FRQ) | | | | | | | | | | | Requisitos no Funcionales (NFR) | | | | | | | | | | |
|-----------|------|------------------------------|------|------|------|------|------|------|------|------|------|------|---------------------------------|------|------|------|------|------|------|------|------|------|--|
| | | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0008 | 0009 | 0010 | 0011 | 0001 | 0002 | 0003 | 0004 | 0005 | 0006 | 0007 | 0008 | 0009 | 0010 | |
| OBJETIVOS | 0001 | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | ↗ | |
| | 0002 | - | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | ↗ | |
| | 0003 | - | - | - | - | - | ↗ | ↗ | ↗ | ↗ | - | - | - | - | - | - | - | - | - | - | - | ↗ | |
| | 0004 | - | - | - | - | - | - | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | - | ↗ | |
| | 0005 | - | - | - | - | - | - | - | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | - | |
| | 0006 | - | - | - | - | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | - | - | - | - | |
| | 0007 | ↗ | ↗ | ↗ | - | - | - | - | - | - | - | - | - | - | - | ↗ | - | - | - | - | - | - | |
| | 0008 | - | - | - | - | - | - | - | - | - | - | - | ↗ | - | - | - | - | - | - | - | - | - | |
| | 0009 | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | - | ↗ | - | - | - | |
| | 0010 | - | - | - | - | - | - | - | - | ↗ | - | - | - | - | ↗ | - | - | - | - | - | - | - | |

Tabla 42: Matriz de rastreabilidad Objetivos-Req. Funcionales-Req. no Funcionales-

3.10 Casos de uso

Esta sección mostrará, a través de diagramas UML, los diferentes requerimientos del sistema, permitiendo tener una visión más clara de los aspectos funcionales de la aplicación, los actores que intervienen y los sistemas externos a los que están conectados.

3.10.1 Gestión de mascotas

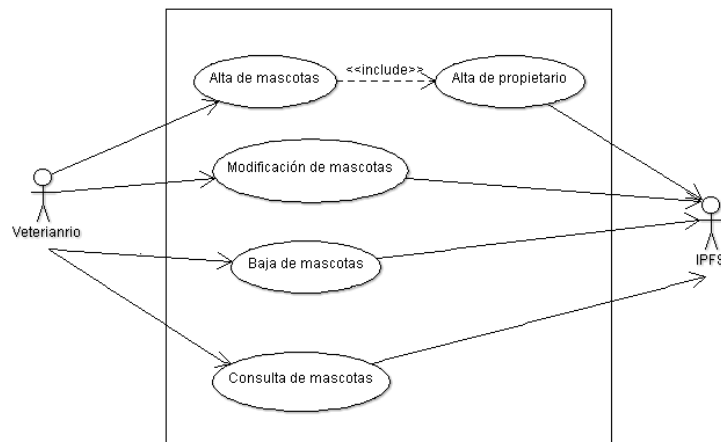


Ilustración 9: Gestión de mascotas

Aquí se muestra como el veterinario podrá realizar las acciones de alta, modificación, consulta y baja sobre los datos de una mascota, y su posterior almacenamiento o consulta en la red IPFS.

| Identificador | UC_001 | |
|------------------|--|--|
| Nombre | Alta de mascotas | |
| Descripción | Almacenará los datos de la mascota en la IPFS. | |
| Precondición | Veterinario registrado en la Blockchain | |
| Postcondición | Mascota almacenada en la IPFS y hash en la Blockchain. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario rellena los datos de la mascota. |
| | 2 | El sistema valida que todos los datos son correctos y que no existen valores sin cumplimentar. |
| | 3 | El sistema envía los datos a la IPFS para su almacenamiento. |
| | 4 | El sistema informa al usuario de que el proceso ha finalizado correctamente. |

Tabla 43: Ficha de caso de uso Alta de mascotas.

| | | |
|-------------------------|---|--|
| Identificador | UC_002 | |
| Nombre | Alta de propietario | |
| Descripción | Almacenará los datos del propietario la mascota en la IPFS. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de la mascota validados | |
| Postcondición | Datos del propietario almacenados en la IPFS y hash en la Blockchain. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario rellena los datos de la mascota. |
| | 2 | Veterinario rellena los datos del propietario. |
| | 3 | El sistema valida que todos los datos son correctos y que no existen valores sin cumplimentar. |
| | 4 | El sistema envía los datos a la IPFS para su almacenamiento. |
| | 5 | El sistema informa al usuario de que el proceso ha finalizado correctamente. |

Tabla 44: Ficha del caso de uso alta de propietario

| | | |
|-------------------------|--|--|
| Identificador | UC_003 | |
| Nombre | Modificación de mascotas | |
| Descripción | Almacenará los cambios realizados en los datos la mascota en la IPFS. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de la mascota existentes en la IPFS. | |
| Postcondición | Datos de la mascota almacenados en la IPFS y hash en la Blockchain. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario busca los datos de la mascota en la IPFS. |
| | 2 | Veterinario modifica los datos de la mascota. |
| | 3 | El sistema valida que todos los datos son correctos y que no existen valores sin cumplimentar. |
| | 4 | El sistema envía los datos a la IPFS para su almacenamiento. |
| | 5 | El sistema informa al usuario de que el proceso ha finalizado correctamente. |

Tabla 45: Ficha del caso de uso de modificación de mascotas

| | | |
|-------------------------|--|--|
| Identificador | UC_004 | |
| Nombre | Baja de mascotas | |
| Descripción | Almacenará los cambios realizados en los datos la mascota en la IPFS. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de la mascota existentes en la IPFS. | |
| Postcondición | Datos de la mascota dada de baja del sistema almacenados en la IPFS y hash en la Blockchain. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario busca los datos de la mascota en la IPFS. |
| | 2 | Veterinario selecciona la causa de la baja de la mascota. |
| | 3 | El sistema valida que todos los datos son correctos y que no existen valores sin cumplimentar. |
| | 4 | El sistema marca a la mascota como dada de baja. |
| | 5 | El sistema envía los datos a la IPFS para su almacenamiento. |
| | 6 | El sistema informa al usuario de que el proceso ha finalizado correctamente. |

Tabla 46: Ficha del caso de uso de baja de mascotas.

| | | |
|-------------------------|--|--|
| Identificador | UC_005 | |
| Nombre | Consulta de mascotas | |
| Descripción | Consulta los datos la mascota en la IPFS. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de la mascota existentes en la IPFS. | |
| Postcondición | Muestra los datos de la mascota almacenados en la IPFS. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario busca los datos de la mascota en la IPFS. |
| | 2 | El sistema muestra los datos de la mascota seleccionada. |

Tabla 47: Ficha del caso de uso consulta de mascotas

3.10.2 Modificación de propietarios

En este Caso de Uso se expone la modificación de los propietarios de las mascotas, que en PetChain siempre se realizará sobre una mascota asociada.

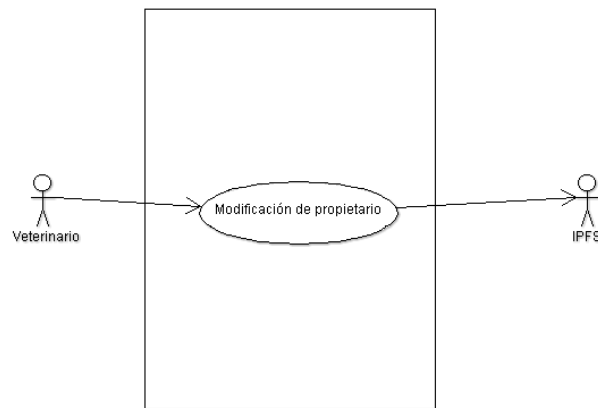


Ilustración 10: Caso de uso modificación de propietarios

| Identificador | UC_006 | |
|------------------|--|--|
| Nombre | Modificación de propietarios | |
| Descripción | Modificación de los datos de un propietario de mascota en la IPFS. | |
| Precondición | <ul style="list-style-type: none"> • Veterinario registrado en la Blockchain. • Datos de la mascota existentes en la IPFS. • Datos del propietario existentes en la IPFS. | |
| Postcondición | Datos del propietario modificados y almacenados en la IPFS. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario busca los datos de la mascota en la IPFS. |
| | 2 | El sistema muestra los datos de la mascota seleccionada. |
| | 3 | El sistema muestra los datos del propietario de la mascota. |
| | 4 | El sistema valida que todos los datos son correctos y que no existen valores sin cumplimentar. |
| | 5 | El sistema envía los datos a la IPFS para su almacenamiento. |

Tabla 48: Ficha del caso de uso modificación de propietarios.

3.10.3 Consulta de saldo

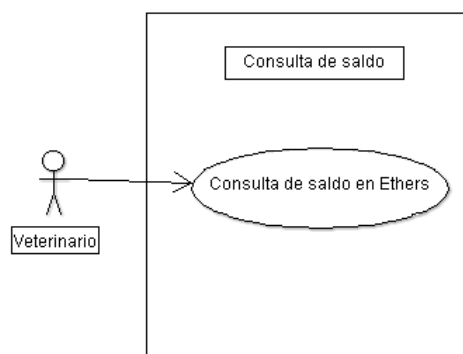


Ilustración 11: Consulta del saldo en Ethers

Este caso de uso muestra la consulta del saldo en moneda Ethers. Cada veterinario recibirá la cantidad de 0,001 ETH de otro veterinario dado de alta en el sistema por las acciones de modificación y baja que este último realice sobre mascotas que el primero introdujo en el mismo.

| | | |
|-------------------------|---|---|
| Identificador | UC_007 | |
| Nombre | Consulta de saldo | |
| Descripción | Consulta los ETH en su cuenta de Ethereum. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de ETH existentes en la IPFS. | |
| Postcondición | Muestra los ETH almacenados en la IPFS, dentro de la pantalla de Transferencias. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario abre la pantalla de Transferencias. |
| | 2 | El sistema muestra el saldo en ETH almacenado en la IPFS. |

Tabla 49: Ficha del caso de uso consulta de saldo

3.10.4 Consulta de transferencias

El veterinario tendrá la posibilidad de realizar la consulta de movimientos en ETH provenientes de las modificaciones realizadas sobre las mascotas que el introdujo en el sistema.

| | | |
|-------------------------|---|--|
| Identificador | UC_008 | |
| Nombre | Consulta de transferencias | |
| Descripción | Consulta las transferencias realizadas en ETH a su cuenta de Ethereum. | |
| Precondición | <ul style="list-style-type: none"> Veterinario registrado en la Blockchain. Datos de transferencias existentes en la IPFS. | |
| Postcondición | Muestra las transferencias realizadas por otros veterinarios procedentes de la modificación de datos de mascotas que fueron identificadas por él, y que están almacenados en la IPFS. | |
| Actores | Veterinario, IPFS | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario abre la pantalla de Transferencias. |
| | 2 | El sistema muestra las transferencias realizadas a favor del veterinario que consulta el sistema y que se encuentran almacenadas en la IPFS. |

Tabla 50: Ficha del caso de uso consulta de transferencias

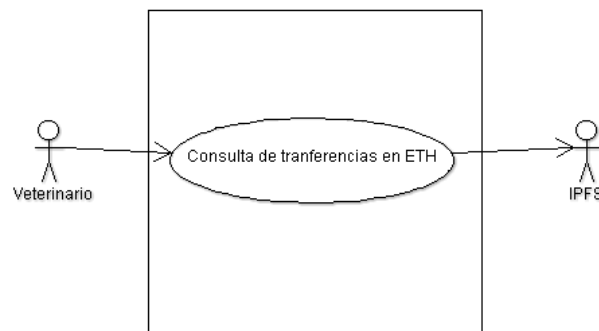


Ilustración 12: Caso de uso consulta de transferencias en ETH.

3.10.5 Registro en la Blockchain

En este caso de uso se describe la necesidad de que un veterinario, antes de poder realizar cualquier acción en la aplicación necesita estar registrado dentro de la Blockchain para que se adjudique una dirección de la red de Ethereum.

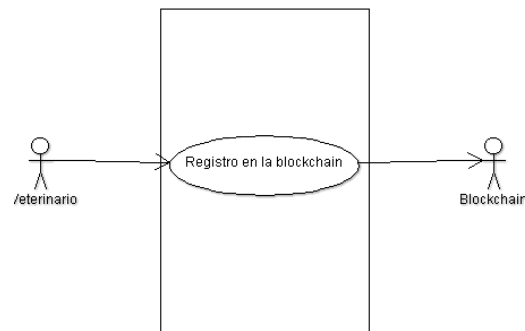


Ilustración 13: Caso de uso de registro en la Blockchain

| Identificador | UC_009 | |
|------------------|---|--|
| Nombre | Registro en la Blockchain | |
| Descripción | El veterinario debe registrarse en la Blockchain para que se le asigne una cuenta de Ethereum. | |
| Precondición | <ul style="list-style-type: none"> Fichero JSON de usuarios creado. Veterinario registrado en el JSON de usuarios de PetChain. Red Ethereum en funcionamiento. | |
| Postcondición | El veterinario tendrá una cuenta de la ethereum asociada y podrá realizar acciones dentro de la aplicación | |
| Actores | Veterinario, Blockchain | |
| Secuencia Normal | Paso | Acción |
| | 1 | Veterinario introduce su nombre de usuario y su contraseña en la pantalla de inicio. |
| | 2 | El sistema muestra una ventana de informativa solicitando el registro mediante su nombre de usuario. |
| | 3 | Si el nombre de usuario es correcto el sistema asigna una cuenta de Ethereum al usuario. |
| | 4 | Si el nombre de usuario no es correcto el sistema vuelve a la pantalla de inicio. |

Tabla 51: Ficha del caso de uso registro en la Blockchain.

4 Descripción técnica.

Petchain es una aplicación web que emplea para su ejecución una combinación de tecnologías, con el objetivo de conectar un frontal en HTML con un backend desarrollado con Smartcontracts e IPFS.

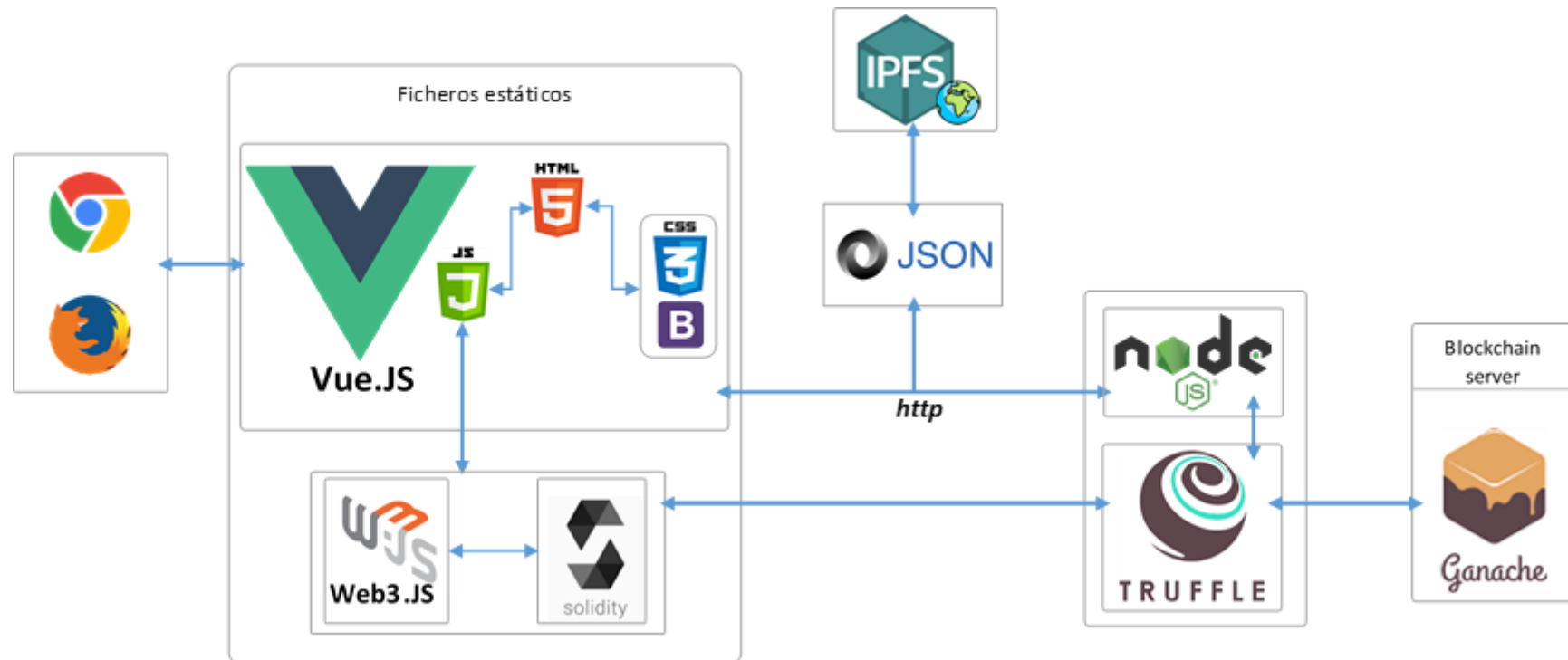


Ilustración 14: Esquema gráfico de herramientas de Petchain (Fuente: diseño propio)

4.1 Arquitectura

Como ya se ha indicado, PetChain es una DApp, diseñada en forma de aplicación web, que tiene entre sus objetivos el mostrar a los usuarios una interfaz con un manejo sencillo y usable, basada en el patrón MVC (*Model View Controller*). “Separa presentación e interacción de los datos del sistema. El sistema se estructura en tres componentes lógicos que interactúan entre sí.” (Sommerville, 2009, pág. 145).

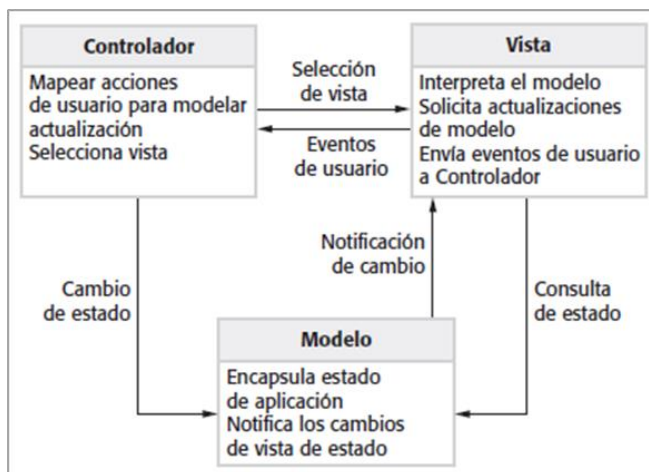


Ilustración 15: Patrón MVC (Fuente: Sommerville, 2009, pág. 156)

El objetivo que se pretende al usar este patrón, es el de lograr una independencia de los elementos que componen la aplicación, con el foco puesto en la simplificación de los procesos evolutivos y correctivos. Para ofrecer una visión más específica de que elementos de la tecnología utilizada forman parte de cada uno de los componentes del patrón, se muestra en la tabla siguiente una correspondencia entre ellos, y en las secciones siguientes se hace una descripción más pormenorizada de los ficheros involucrados dentro de cada una de las capas del patrón MVC.

| Relación Arquitectura- tecnologías utilizadas | |
|---|------------|
| Vista | Vue (html) |
| | css |
| | JavaScript |
| Controlador | Javascript |
| Modelo | Solidity |
| | JSON |

Tabla 52: Relación Arquitectura - Tecnología

4.1.1 Modelo

Es la capa que se comunica con los datos, bien con la base de datos o con una lógica del negocio, y cuyo objetivo es desacoplar la parte gráfica de los datos que maneja la aplicación. En Petchain la capa que conforma el modelo está compuesta, por el conjunto de ficheros *.sol (Solidity), los ficheros con estructuras JSON que se emplean para poblar elementos del frontal y los archivos Javascript que componen los servicios, así como los necesarios para la compilación de los ficheros Solidity que se localizan en la carpeta /truffle/migrations.

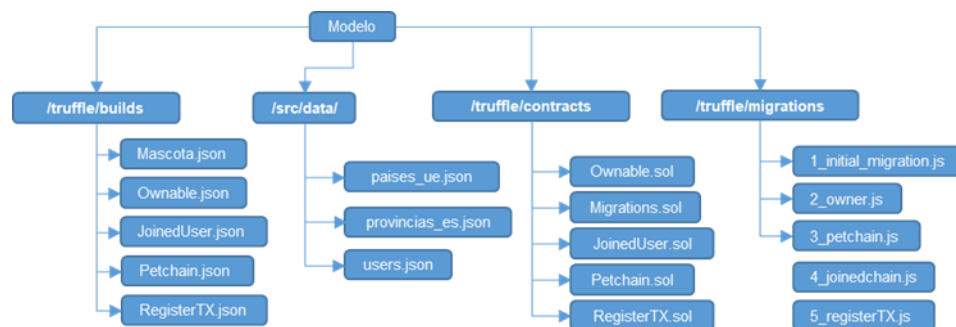


Ilustración 16: Estructura de los ficheros del Modelo

4.1.2 Vista

La vista es el conjunto de ficheros que componen la parte gráfica y que determinan como debe de visualizarse la aplicación. El conjunto de ficheros que componen la estructura de la vista son los reflejados en la imagen siguiente.

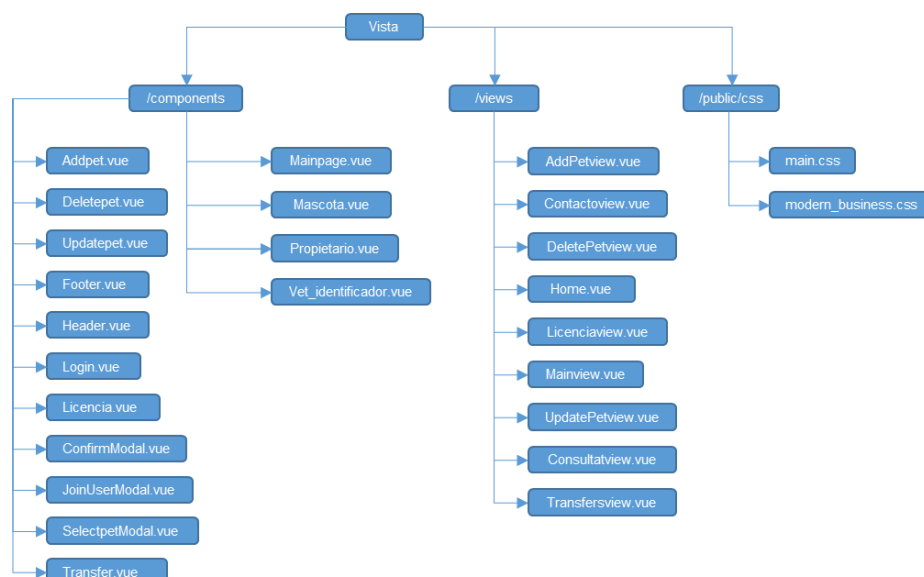


Ilustración 17: Estructura de ficheros de la Vista

Obsérvese la particularidad de la no existencia de ficheros *.html en la estructura mostrada. En su lugar están los ficheros con extensión *.vue, tanto para los componentes como para las

vistas que componen la aplicación. Esta característica se comentará en secciones posteriores y en profundidad, pues está íntimamente ligada al framework Vue.js que ha utilizado para el desarrollo del proyecto.

4.1.3 Controlador

Esta es la capa encargada de realizar las modificaciones realizadas por el modelo y/o la vista en respuesta a las peticiones realizadas desde el frontal o la base de datos. Su objetivo es servir como elemento de abstracción entre el modelo y la vista. En la aplicación se ha buscado optimizar la codificación, es decir hacerla lo más estándar posible.

La labor de esta capa es de suma importancia, pues en ella se genera y se envía o se recibe un objeto con notación JavaScript (JSON), de los datos procedentes de la interfaz de usuario, o bien de los procesados por la lógica almacenada en los ficheros Solidity y que es enviada a la IPFS.

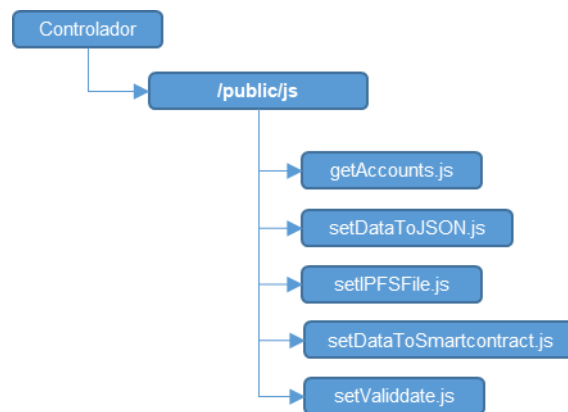


Ilustración 18: Esquema de la capa controladora

4.1.4 Análisis de elementos: Componentes y Vistas.

Hasta el momento se ha realizado una exposición del patrón que se ha seguido para la arquitectura del proyecto. Hemos visto que elementos forman cada una de sus partes y se ha destacado en el apartado 5.1.2, la particularidad de la no existencia de ficheros *.html dentro de su estructura. En su lugar hay ficheros con la extensión *.vue alojados en dos carpetas: /components y /views. Los nombres de cada una de ellas son suficientemente descriptivos, en la primera se alojan los componentes, que serían los ladrillos básicos que forman las pantallas, y en la segunda se alojan las vistas, que son agrupaciones de 1 o n componentes y que tienen como finalidad su presentación en pantalla.

Por otro lado, la agrupación de componentes en vistas, facilita su comunicación a la hora de realizar flujos de datos entre ellos o realizar algún tipo de cambio de estado debido a que la vista actúa como “padre” y los componentes como elementos “hijo”, permitiendo una comunicación bidireccional. Esta característica reduce el número de ficheros empleados así como las líneas de código necesarias dentro de la aplicación.

En las subsecciones siguientes se realiza una descripción más pormenorizada de estos dos elementos.

Componentes

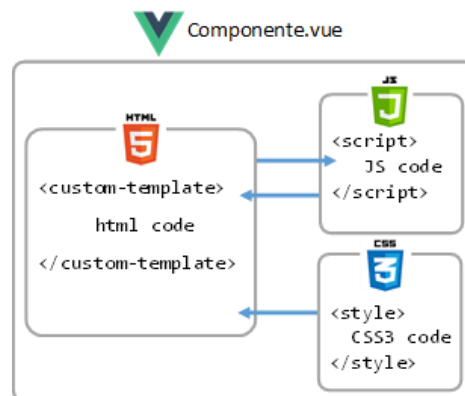


Ilustración 19: Elementos de un componente Vue.js

Como se ha introducido en la sección anterior, los componentes son los ladrillos que forman las pantallas de nuestra aplicación.

En la imagen anterior se observa de manera gráfica la estructura de un componente. En ella se distinguen tres elementos:

- **<template>**: En esta sección se aloja el código html que forma el componente y que se visualizará en el navegador.
- **<script>**: Sección que contiene el código JavaScript necesario para el funcionamiento del componente. Aquí es donde se importan ficheros externo, se definen propiedades, los valores que toman estas y que se representan posteriormente en el template gracias a la comunicación bidireccional existente entre ellos.
- **<style>**: Sección que contendrá los estilos css que manejarán el formato de presentación del componente en pantalla. A diferencia de los dos anteriores, su definición no es obligatoria, pues los estilos pueden ser establecidos en ficheros externos a nivel general de la aplicación pero si es conveniente, en caso de necesitarlos, para favorecer la encapsulación de comportamientos.

El uso de componentes tiene unos beneficios directos en relación al código y al diseño:

- **Reutilización**: Un componente sólo es necesario definirlo una vez y puede ser utilizado en tantas vistas como sea necesario.
- **Encapsulación**: El comportamiento de un componente está restringido a su ámbito de actuación o “scope”, sin que interfiera, si así se requiere, con el resto de código de una vista.

Vistas

Las vistas son agrupaciones de 1 o n componentes que combinándolos e interactuando entre ellos, permiten la definición de las pantallas. A nivel de código tienen la misma estructura que éstos, pero incorporan una característica muy importante que los hacen diferentes, y es que importan en su interior los ficheros necesarios para la formación de las pantallas.

En la imagen siguiente se muestra el flujo de composición de la pantalla de Contacto (/src/views/Contactoview.vue).



Ilustración 20: Flujo de composición de la vista `Contactoview.vue`

1. En primer lugar se importan los ficheros que van a componer la vista asociándolos a un "alias". En nuestro ejemplo importamos `Menu.vue` y `Contacto.vue`, asociándolos a los alias "Menu" y "Contacto", respectivamente.
2. A continuación indicamos que esos dos elementos los exportamos y que se van a comportar como componentes. Al hacer esto, le estamos indicando al framework que vamos a utilizar etiquetas personalizadas, con formato html y que en su interior van a contener todo el código de cada uno de los elementos importados.
3. En la sección del `template` ahora podemos definir tantas etiquetas personalizadas como sea necesario, con el objetivo de formatear la pantalla que necesitamos.
4. Por último, y tras la compilación de los ficheros, el navegador es capaz de representar las pantallas con el formato que hayamos definido, en nuestro ejemplo la pantalla de Contactos.

4.2 Arquitectura técnica.

Hasta ahora se ha analizado el patrón arquitectónico de PetChain relacionándolo con los componentes de la aplicación, con el objetivo de tener una idea clara de la relación entre ellos. Ahora se pretende ahondar en la arquitectura técnica, definiendo y describiendo las capas que la componen así como la relación entre ellas.

Para comenzar, en la siguiente ilustración, se muestra una descripción detallada de los elementos, su posición en las capas de la aplicación y su forma de relacionarse.

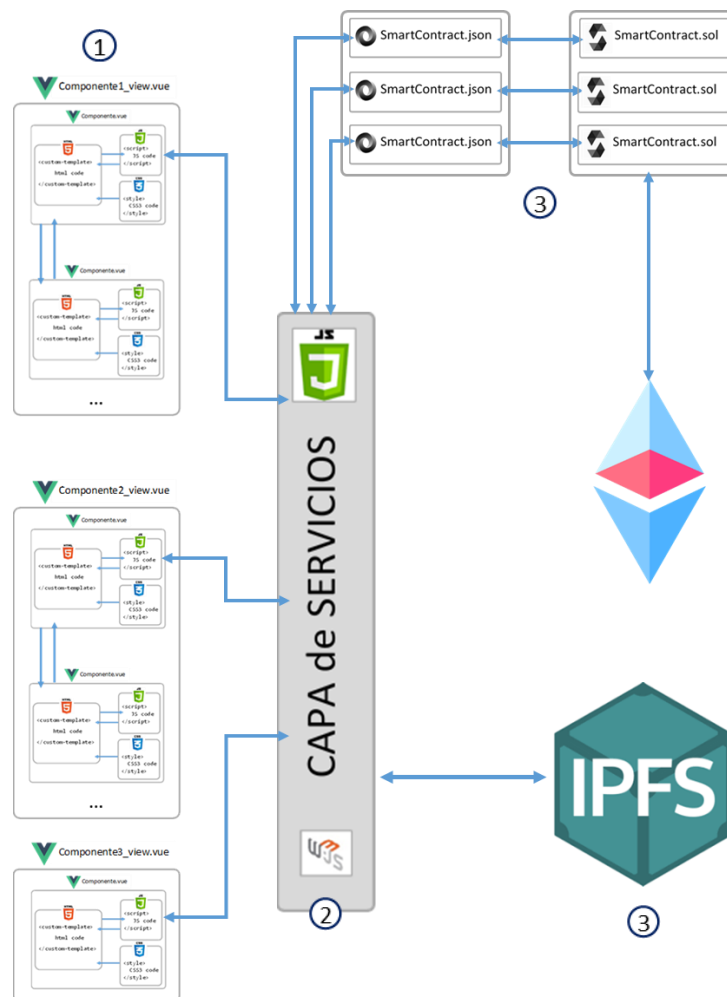


Ilustración 21: Esquema de componentes

1. **Capa de presentación:** Formada por los componentes, y estos a su vez agrupados en las vistas, estos elementos ya han sido descritos en secciones anteriores. Como se puede observar todos los componentes se relacionan con el exterior a través de la sección del *script*. Desde aquí se pueden hacer invocaciones a los elementos padres, hijos o hermanos, así como a otras vistas. En nuestra arquitectura, existe una comunicación entre componentes de la misma y de diferentes vistas, y también con la capa de servicios. Todas las comunicaciones con esta capa se realizan siempre invocando o recibiendo objetos de tipo *data()*, en formato JSON. De este modo se evita el acoplamiento y se facilita el mantenimiento evolutivo y correctivo.
2. **Capa de servicios:** Es la encargada de comunicar la capa de presentación con el backend y viceversa. Aquí se producen las comunicaciones bidireccionales entre los Smartcontracts para su ejecución en la red Ethereum, con el servidor Ganache

y el almacenamiento y recuperación de ficheros almacenados en la IPFS. Está formada por los ficheros:

- a. `/public/js/services/getAccounts.js`: recibe las cuentas del servidor Ganache, para asociarlas a los usuarios del sistema.
- b. `/public/js/services/setIPFSFile.js`: encargado de subir a la IPFS el objeto JSON con los datos de las mascotas y propietarios, así como de recuperar el hash generado en el almacenamiento.
- c. `/truffle/builds/`: En esta carpeta se alojan los ficheros `*.json`, que son el resultado de la compilación de los Smartcontracts codificados en lenguaje Solidity (`*.sol`). Éstos son con los que el framework Web3.js es capaz de interactuar, y por lo tanto que se realice la comunicación con ellos desde la capa de front-end.

3. **Capa de backend**: Formada por:

- a. La red Ethereum, que en nuestro caso la ofrece el servidor local Ganache en la url <http://127.0.0.1:7545>.
- b. La IPFS servida desde el nodo local desplegado en la url: <https://127.0.0.1:5001>

Esta capa es la que dota a PetChain de un cierto aspecto novedoso y su importancia en el conjunto de la aplicación va ser tratado en profundidad en la siguiente sección.

4.3 Arquitectura del backend

En el conjunto del proyecto se quiere destacar la metodología empleada para dotarlo de una capa de persistencia de datos, sin que para ello haya sido necesario el utilizar una base de datos “tradicional”.

Como ya se ha introducido anteriormente, PetChain hace uso de IPFS para servir como soporte para sus datos. Este protocolo convierte toda la información que le llega en un hash, y es esta forma de trabajar la que se ha aprovechado para diseñar un sistema que, junto con las estructuras o “structs” de los Smartcontracts, permiten el almacenamiento de la información.

La técnica que se ha desarrollado hay que dividirla en dos pasos, el proceso de almacenamiento y el de la recuperación de la información en la IPFS.

4.3.1 Almacenamiento en la IPFS

Partimos del punto que el frontal, es una aplicación web tradicional, y como tal nos va permitir el recoger toda la información y almacenarla de manera estándar. En el proyecto se ha optado por convertirla en un objeto JSON estándar.

En el primer paso del proceso de Alta, se genera un objeto que contiene toda la información correspondiente al veterinario identificador, la mascota y su propietario, con la siguiente estructura de campos:

- Veterinario identificador:

```
"vetidentificador": {  
  "vetName": "Itziar",  
  "vetId": 12345,  
  "vetSurname": "García de la Plaza",  
  "vetCol": "Avila"  
}
```

- Mascota:

```
"mascota": {  
  "ultima_rev": [],  
  "petIdNumber": "98756-AA",  
  "fechImplantacion": "2019-08-24",  
  "fechNac": "2019-08-01",  
  "fechalta": "2019-08-24",  
  "petName": "Chula",  
  "raza": "pura",  
  "razaPet": "Dogo aleman",  
  "razaOtra": null,  
  "danger": "dangerNo",  
  "capa": "Blanco",  
  "pelo": "Corto",  
  "genero": null,  
  "passport": "1234567B",  
  "aptitud": "Compania"  
}
```

- Propietario:

```
"propietario": {  
  "propName": "Carlos",  
  "propSurname": "Escario Bajo",  
  "propId": "1236",  
  "propPhone": "918960203",  
  "provProp": "AV",  
  "ciudadProp": "S.L. de El Escorial",  
  "cp": "28200",  
  "propPais": "ES"  
}
```

En un segundo paso se almacena este objeto en la IPFS. Este proceso se realiza en la capa de servicios dentro del fichero *public/js/services/setIPFSFile.js*. Este es el encargado de enviar el objeto generado al nodo local de IPFS situado en <https://127.0.0.1:5001>. Aquí se va a realizar la conversión de la información enviada a un hash, en concreto el objeto anterior está convertido en: *Qmck1hAc9sYTHJJZcr68HxdHLCPMFgSn9QMSSar7j85TFg5*⁸

⁸ Objeto disponible en: <https://ipfs.infura.io/ipfs/Qmck1hAc9sYTHJJZcr68HxdHLCPMFgSn9QMSSar7j85TFg5>

Puede parecer obvio, pero no es fácil recordar este tipo de información, pero si el del identificador de la mascota, que se encuentra almacenado en el chip que lleva incorporado. Por lo tanto el tercer, y último paso es el de relacionar en una estructura externa este identificador con el hash. Esto se produce en el Smartcontract *Petchain.sol*, que se encuentra desplegado en la Blockchain. Aquí vamos a encontrar una estructura o “struct”, que relaciona el identificador de la mascota con el hash que se acaba de guardar en la IPFS.

De esta manera, tenemos la información almacenada de manera ligera y a la vez consistente. En el esquema siguiente se observa este proceso de manera simplificada.

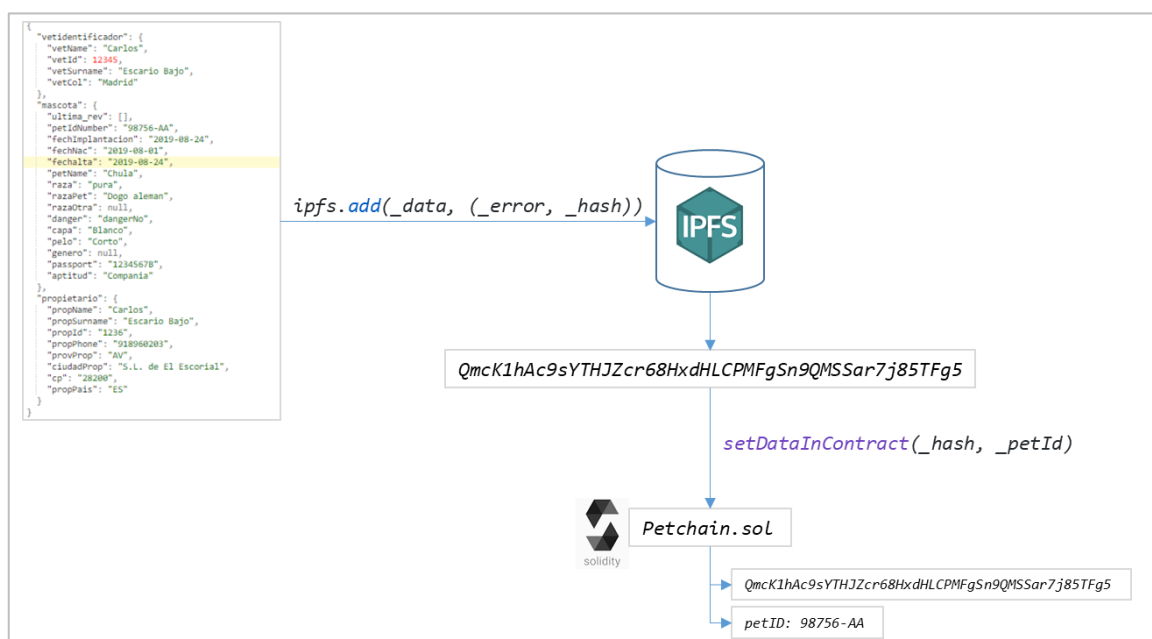


Ilustración 22: Esquema de generación y almacenamiento de la información.

4.3.2 Recuperación de información

El proceso de la recuperación se produce en base a la premisa de conocer o de tener localizado el identificador de la mascota. Éste valor lo vamos a encontrar, o bien en cualquier documento o en el chip que todo perro o gato deben de llevar implantado. Con este código es posible recuperar la información correspondiente al animal para su modificación o baja.

El proceso que sigue el sistema en este caso es el inverso. En el primer paso si se introduce este identificador en Petchain, se realiza una llamada al Smartcontract *Petchain.sol* que, a través de este valor busca en la estructura a la que nos referimos en el punto anterior el ítem que tiene almacenado. Si encuentra el valor devolverá el valor del hash que tiene asociado.

Con el valor del hash obtenido, el segundo paso consiste en construir la url e ir a la IPFS a buscar el objeto asociado a ese valor. Si el objeto existe se devolverá, y sus valores se podrán cargar en el modelo de la aplicación y poder a realizar los cambios que se necesiten.

En la imagen siguiente se puede observar un esquema del proceso descrito.

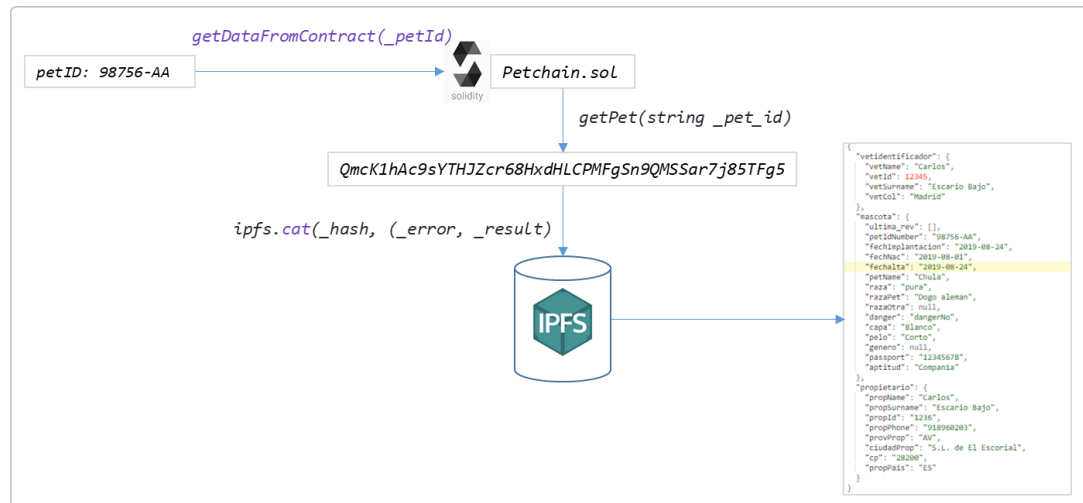


Ilustración 23: Esquema de recuperación de la información de la IPFS

4.4 Entorno de desarrollo

Todos los elementos anteriores necesitan estar orquestados por diversas tecnologías, que deben de trabajar en conjunto. En los próximos apartados se van a introducir y explicar con detalle cada una de las herramientas, frameworks y servidores necesarios para el correcto funcionamiento de la aplicación.

4.4.1 Node.js

Node.js es un entorno multiplataforma de código abierto, disponible para su descarga libre en <https://Node.js.org>, y utilizado para la ejecución de JavaScript en el lado del servidor. Su funcionamiento se basa en la ejecución de eventos de manera asíncrona, esto significa que mientras Node.js gestiona cualquier solicitud de I/O que le haya llegado y hasta que la petición finalice, ésta se estará ejecutando en segundo plano, y mientras el sistema podrá seguir realizando cualquier otra operación en primer plano.

Gran parte de su éxito se basa en la enorme cantidad de módulos existentes que permiten extender las funcionalidades del entorno a través del gestor de paquetes **npm** (Node Package Manager)⁹.

En el proyecto estamos utilizando la versión 8.12.0, que aunque no es la más actualizada para su descarga (a la fecha de la redacción del presente trabajo la última versión disponible es la 10.16), cumple completamente con las necesidades del mismo, pues si es requisito

⁹ npm web site: <https://www.npmjs.com/>

técnico para el funcionamiento de los módulos instalados, que la versión de Node.js sea superior a la 6.0.0.

Dependencias

Los módulos a los que nos hemos referido anteriormente, y que son necesarios para el funcionamiento de PetChain, son los que hay definidos en el fichero */package.json* dentro del objeto *dependencies*.

- **core-js**: librería de utilidades Javascript.
- **ipfs-http-client**: módulo con un API que nos permitirá conectar con la IPFS.
- **jquery**: versión 3.4.1 de la conocida librería de javascript.
- **vue**: módulo que contiene las librerías necesarias del framework Vue.js.
- **vue-router**: utilidad para la gestión de url's y navegación entre pantallas.
- **web3**: módulo que contiene las librerías necesarias del framework Web3.js. La versión instalada es la 0.20.6, pues para versiones posteriores hay reportadas incompatibilidades con el servidor de Truffle empleado para la compilación de los ficheros Solidity y el servidor Ganache. Por lo que tras múltiples consultas en foros de internet se ha optado por instalar esta versión, que aunque no es la más actualizada si es la más estable hasta la fecha.

4.4.2 Vue.js

Para la generación del proyecto se ha escogido el framework de Javascript Vue.js. Éste junto con Angular y React, forman la trilogía más utilizada a nivel global para el desarrollo de aplicaciones SPA bajo Node.js.

Se ha escogido Vue.js fundamentalmente por diversas razones:

- **Curva de aprendizaje**: Previamente al desarrollo del presente proyecto sólo se tenían conocimientos de Angular no así de Vue.js ni de React. La experiencia con Angular no había sido muy positiva, pues la exigencia de la curva de aprendizaje era muy alta. Tras la primera toma de contacto con Vue y React, se ha observado la sencillez de uso del primero, pudiendo comprobarlo con la creación de ejemplos sencillos de SPA's, en contraposición a React, que de inicio exige el aprendizaje de una nueva sintaxis de etiquetas denominada JSX.
- **Documentación**: Vue.js contiene una muy extensa documentación para su aprendizaje, tanto en texto como en vídeo, dentro de su web: <https://vuejs.org/v2/guide/>. Asimismo, existe una muy amplia documentación en castellano que ayuda a un mejor aprendizaje.
- **API**: el API de Vue.js es realmente sencilla e intuitiva.

- **Creación de proyecto:** La creación de un proyecto Vue.js se realiza con un solo comando en la ventana del sistema, generando automáticamente el *scaffolding* del proyecto, sin necesidad de arduas configuraciones. PetChain ha sido desarrollado con la librería Vue-CLI, que está instalada como una dependencia en el entorno de Node.js, y permite la generación de un proyecto web en un entorno local.
- **Detección de errores en el puerto 8080:** Vue.js se ejecuta por defecto en modo desarrollo dentro de un servidor local en la url <http://localhost:8080>. El framework tiene la capacidad de detectar en tiempo de compilación si el puerto 8080 tiene algún problema y de manera automática redirecciona la url de ejecución al puerto 8081, 8082, etc., e informando de ello al usuario vía consola para su ejecución en el navegador.

Requerimientos

- Versión de Node.js +8.9.0
- Versión de npm +6.0

La instalación de Vue-CLI la realizamos con el siguiente comando:

```
> npm install -g @vue/cli
```

Tabla 53: Comando de instalación de Vue-CLI

4.4.3 Truffle

Truffle¹⁰ es un framework que permite el desarrollo, compilación, despliegue y pruebas de Smartcontracts sobre redes Ethereum. Es una herramienta que se instala como módulo de Node.js, v8.9.4 o superior, y está disponible para las plataformas Windows, Linux y Mac OS X. Las características que ofrece son:

- Scaffolding inicial del proyecto.
- Compilación de Smartcontracts y generación de archivos en formato JSON con la estructura del Smartcontract y su bytecode correspondiente.
- Despliegue sobre redes privadas y públicas.
- Generación y ejecución de test.

Instalación de Truffle

Como se ha comentado anteriormente Truffle está disponible como un módulo del entorno de ejecución Node.js, por lo tanto para su instalación haremos uso de npm (Node Package Manager). Ejecutamos entonces la sentencia siguiente:

¹⁰ Truffle website: <https://truffleframework.com>


```
> npm install -g truffle --save-dev
```

Ilustración 24: Comando de instalación de Truffle

Los parámetros de instalación son opcionales, pero permitirán una ejecución y actualización de las dependencias, en caso de actualizarse las versiones.

Generación del proyecto

Dado que hemos realizado la instalación de Truffle con el parámetro `-g`, desde ahora vamos a tener disponible desde cualquier parte de la estructura de archivos de nuestra máquina los comandos ejecutables del framework, que permitirán su instalación y configuración desde la línea de comandos.

Para nuestro proyecto, ejecutaremos:

```
> truffle init
```

Ilustración 25: Comando de inicialización de un proyecto Truffle

Este comando genera la estructura necesaria para el proyecto, que nos permitirá el almacenamiento de los ficheros Solidity, su compilación y despliegue de binarios. Dicha estructura consta de los siguientes elementos.

- `contracts/`: directorio para el almacenamiento de ficheros `*.sol`.
- `test/`: directorio para el almacenamiento de los test.
- `migrations/`: directorio para el almacenamiento de los scripts correspondientes a las migraciones de los archivos con extensión `*.sol` a ficheros `*.js`, y que contendrán la estructura del Smartcontract en formato JSON, así como en bytecode.
- `truffle-config.js`: fichero de configuración de Truffle para la definición de las redes de despliegue de los Smartcontracts.



ATENCIÓN: Es necesario indicar que en la instalación de Truffle, se requiere **obligatoriamente** que la carpeta en donde se vaya a ejecutar el comando `truffle init`, esté completamente **vacía**. Si no se cumple esta premisa, el framework devolverá siempre un error indicando que no es posible su instalación para evitar sobre-escrituras a proyectos ya existentes.

4.4.4 Ganache

Ganache¹¹ es un componente multiplataforma (Windows, Linux o Mac OS X), que forma parte del framework Truffle y su función es la trabajar como servidor de cuentas en una red

¹¹ Ganache website: <https://truffleframework.com/ganache>

Ethereum. Es el complemento ideal para el desarrollo de aplicaciones Blockchain en entornos cerrados, locales o corporativos no productivos o de desarrollo, y está disponible para su ejecución como herramienta de escritorio o para invocarlo a través de línea de comandos (TestRPC).

Instalación de Ganache

La instalación de Ganache es trivial y se realiza a través de un clásico “wizard”. Para el presente proyecto se implementará como una herramienta de escritorio, tanto para un sistema

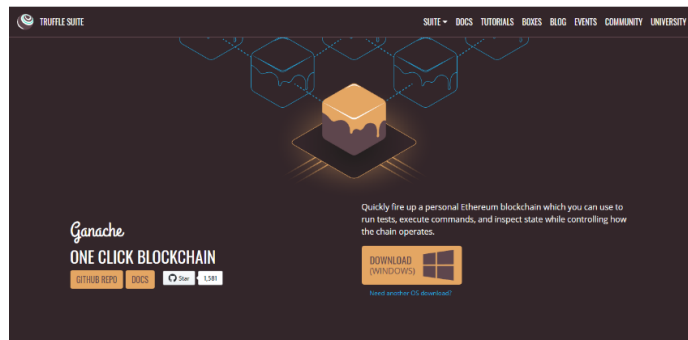


Ilustración 26: Web de Truffle framework.

Windows 10 y Linux Ubuntu 18.02, a través del instalador que se puede descargar desde la página web de Truffle.

4.4.5 Web3.js

Web3.js es la herramienta JavaScript, open source, multiplataforma e imprescindible para interactuar desde el navegador contra nuestro servidor Ethereum a través del protocolo http. Este servidor, será el que se denomine *object provider*, que en nuestro caso será el servidor de cuentas Ganache. Para que exista esta comunicación entre el frontal web y el *object provider*, web3.js expone un API¹² con gran cantidad de métodos con los que comunicarse con su entorno.

Instalación de Web3.js

Esta librería está disponible de diversos métodos:

- Es posible incorporarla a nuestro proyecto como un módulo de Node.js a través del comando npm en la carpeta del proyecto, y se instalará como una dependencia del proyecto dentro del package.json de Node.js. Como se ha indicado en el punto 5.2.1, con el fin de evitar incompatibilidades con Truffle la versión instalada es la 0.20.6.

¹² <https://github.com/ethereum/wiki/wiki/JavaScript-API#web3-javascript-app-api-for-02xx>

- También es posible tener disponible la librería cuando se instala el framework de Truffle, dado que está incorporado como parte del paquete., la cual tiene bastantes problemas en cuanto a funcionalidades deprecadas.

```
> npm install -g web3 --save-dev
```

Tabla 54: Instalación de web3.js

4.4.6 Bootstrap

Bootstrap¹³, framework original de la empresa Twitter, es quizás una de las librerías de código abierto más conocidas en el sector del desarrollo web. Permite crear webs, webapps y DApps adaptables (*reponsive design*), multiplataforma, bajo el paradigma de “mobile first”. Con una amplísima documentación a lo largo de toda la Internet, la correcta aplicación de su extensa gama de plantillas gratuitas, estilos y scripts predefinidos, permiten al desarrollador el diseño de pantallas de diseño limpio y claro, con estilos modernos y con una extensa gama de componentes.

Desde contenedores, menús desplegables, formularios, validadores, componentes HTML, etc., Bootstrap ofrece un ahorro en tiempo y código a los diseñadores y programadores web, y, que bajo el criterio del autor, no hay duda en calificarla como la mejor herramienta existente en la actualidad, dentro de su dominio de aplicación.

La implementación dentro de Petchain se ha realizado descargando los ficheros estáticos de la web del framework, e incorporándolos a la estructura del proyecto dentro de la carpeta */src/vendor/bootstrap*. Para la aplicación se descarga la última versión disponible, la 4.3.1.

La librería de Bootstrap la incorporamos a través de un CDN de acceso público, lo que nos proporciona la ventaja de:

- Ahorro de espacio en el proyecto, pues nos evita la descarga de ninguna clase de fichero en el directorio local.

La invocación de los ficheros necesarios para tener accesible Bootstrap se realiza en */public/index.html* a través de las siguientes urls:

- <https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css>: para el minificado de los estilos css.
- <https://maxcdn.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js>: para el minificado con el javascript necesario para interactuar con los estilos Bootstrap.

¹³ <https://getbootstrap.com/>

4.5 Herramientas utilizadas

4.5.1 Argo UML

Herramienta Open Source, desarrollada en Java, bajo licencia EPL y con soporte para diagramas UML ver, 1.4. Ha sido el software utilizado para el diseño de los diagramas de Casos de Uso y los Diagramas de secuencia. Se encuentra disponible para su descarga gratuita en: <http://argouml.tigris.org/>.

4.5.2 REM

Según se cita en la web del Departamento de Lenguajes y Sistemas Informáticos de la Universidad de Sevilla (2004)¹⁴:

REM (REquirements Management) es una herramienta experimental gratuita de Gestión de Requisitos diseñada para soportar la fase de Ingeniería de Requisitos de un proyecto de desarrollo software de acuerdo con la metodología definida en la Tesis Doctoral "Un Entorno Metodológico de Ingeniería de Requisitos para Sistemas de Información", presentada por Amador Durán en septiembre de 2000.

4.5.3 GitHub Desktop

Cliente gráfico de GitHub, para la conexión al repositorio remoto de Git. Herramienta gratuita y disponible para su descarga en <https://desktop.github.com/>. Se ha utilizado como herramienta de almacenamiento en el repositorio tanto para el código del proyecto Petchain, como para la presente memoria.

4.5.4 Trello

Software para la gestión de tareas de un proyecto, basado de una interfaz web, y que permite de modo visual llevar el control de éstas en formato de tableros con tarjetas, en donde se podrán definir, listas, comentarios, fechas de entrega, miembros de la tarea, etc. Desde el año 2017 forma parte de la lista de herramientas de la empresa Atlassian. Disponible en <https://trello.com/>, dispone de licencias gratuitas y premium.

4.5.5 Visual Studio Code

Editor de código fuente de la empresa Microsoft, con múltiples capacidades. Disponible para las plataformas Windows, Linux y Mac OS, permite además de la edición de código, conexión con repositorios Git, depuración de código, resaltado de errores de código y con una

¹⁴ http://www.lsi.us.es/descargas/descarga_programas.php?id=3

extensa lista de lenguajes soportados como Javascript, Solidity, C, C++, C#, Java, Python, PHP, Go, etc. Disponible para su descarga gratuita en: <https://code.visualstudio.com/download>.

5 Petchain. Descripción funcional

En el capítulo anterior se introdujo de una manera detallada las características técnicas de la aplicación mostrando una visión a bajo nivel de la misma. En el presente, el objetivo es que el lector obtenga una visión en profundidad de los aspectos funcionales de la aplicación con el objeto de que se familiarice con su manejo y funcionamiento.

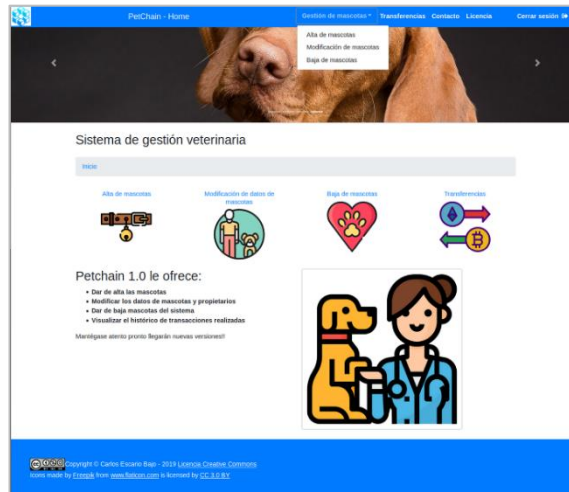


Ilustración 27: Pantalla principal de PetChain

5.1 Login

En correspondencia al FRQ-001¹⁵, el acceso a la aplicación está securizado, mediante usuario y contraseña a través de la pantalla de login, en donde el usuario debe de estar registrado previamente en el fichero /src/data/user.json. En caso de no introducirse las credenciales correctamente el sistema lo indicará con un mensaje de “Usuario no encontrado”, en color rojo y situado la zona superior del formulario de entrada.

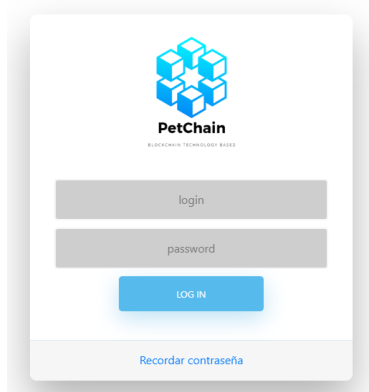


Ilustración 28: Login PetChain

¹⁵ Sección 3.8.2. Requisitos funcionales

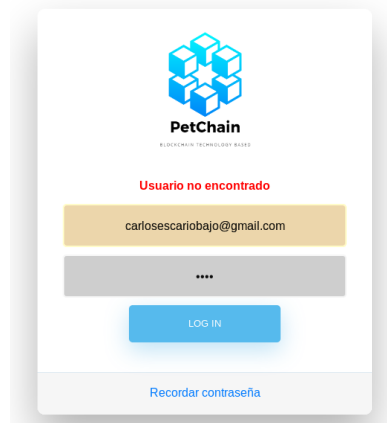


Ilustración 29: Login incorrecto.

Tal y como muestra la imagen anterior, en caso de no recordar la contraseña, en la zona inferior de ventana de login existe un enlace que permite al usuario la posibilidad de solicitar al sistema, a través del envío de un correo electrónico a la dirección que introduzca, un recordatorio con la contraseña que tiene registrada en el sistema.

La ventana muestra un campo “input” de tipo “email”, que no necesita validación adicional a la que ofrece HTML5, de modo que mientras el valor introducido no cumpla con las reglas que se establecen por defecto, el formulario no enviará el correo para la recuperación de la contraseña.

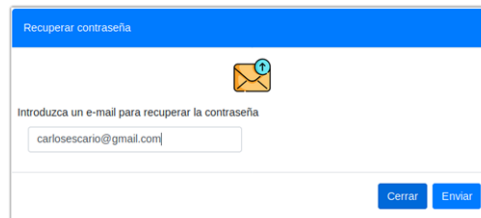


Ilustración 30: Ventana recordatorio de contraseña.

5.1.1 Registro en la Blockchain

En correspondencia al requisito FRQ-0012¹⁶ la aplicación solicitará al usuario, la primera vez que ingresa en la pantalla principal de Petchain su registro en la cadena de bloques. Esta acción se realizará a través de una ventana modal que impedirá seleccionar cualquier opción de navegación, hasta que el usuario ingrese en el formulario que se muestra, la dirección de correo electrónico con la que ha accedido al sistema.

¹⁶ Sección 3.8.2. Requisitos funcionales

La acción de cerrar la ventana, redirigirá al usuario de nuevo a la ventana de Login inicial, impidiendo la navegación por el sistema.

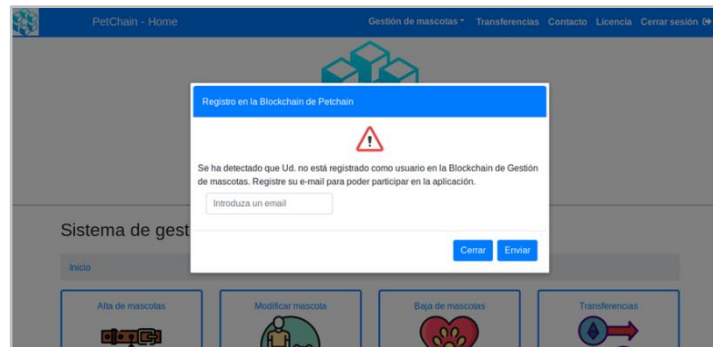


Ilustración 31: Ventana de registro en la Blockchain.

Señalar por último que una vez realizado el registro correctamente, esta ventana emergente ya no se mostrará en lo sucesivo mas veces a este usuario.

5.2 Pantalla principal

La pantalla principal es, como en cualquier tipo de aplicación web, una recepción al usuario, y en ella se agrupan las funcionalidades ofrecidas, tanto en formato de enlaces en el área central, como en el menú que se sitúa en la zona superior de la aplicación y que estará presente durante todo el proceso de navegación.

Tal y como se muestra en la imagen que ilustra el inicio del presente capítulo se ha pretendido diseñar una pantalla sencilla, evitando contrastes de colores muy fuertes y con imágenes en las que dominen las líneas sencillas y los colores pastel, con el objeto de evitar al usuario una “fatiga visual”. Por otro lado, se han buscado imágenes, siempre dentro del catálogo existente, que tengan cierta relación visual con el texto de enlace y la opción a la que va a navegar.

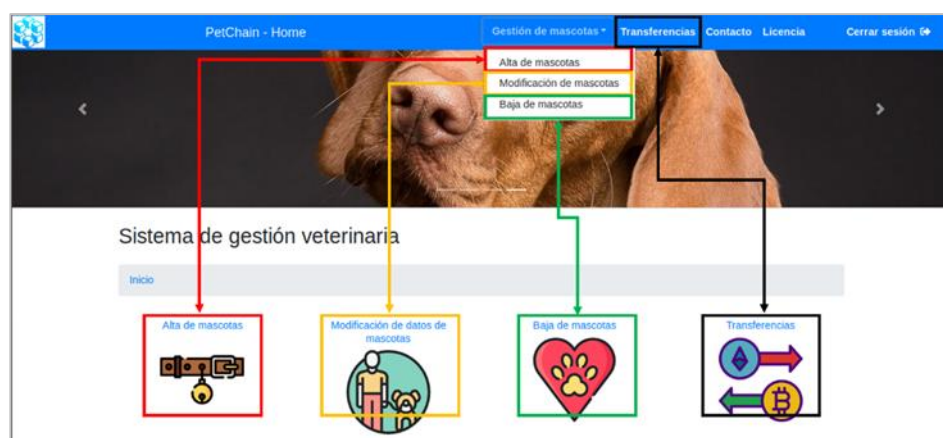


Ilustración 32: Opciones de navegación

De este modo, tal y como representa la imagen anterior, en la zona central se encuentran las opciones principales de navegación dentro de la aplicación, que están directamente relacionadas con las existentes en el menú superior. Aunque en este paso se encuentren duplicadas, durante la fase de diseño se consideró que dado que esta duplicidad estaba sólo en este punto, estos elementos eran lo suficientemente representativos como para que no supusiese un problema de usabilidad para el usuario.

5.2.1 Menú de navegación

Como ya se ha comentado estará presente durante toda la navegación, pues las opciones centrales de alta, modificación, consulta, baja de mascotas y las transferencias con sus imágenes asociadas sólo van a estar visibles en esta pantalla, en el resto de la aplicación desaparecen.

Se ha definido un menú simple en opciones y con comportamiento 100% responsive, gracias al uso del framework Bootstrap 4, que establece este tipo de comportamiento de manera muy sencilla de implementar.

5.2.2 Footer (pie de pantalla)

Asimismo, en la zona inferior se sitúa el “footer” o pie de pantalla, que al igual que el menú también estará presente durante todo el proceso de navegación, pues en el se sitúan dos elementos que pueden pasar desapercibidos para el usuario, pero que de cara al desarrollo del presente trabajo son de considerable importancia.

- Información del tipo de licencia Creative Commons seleccionada: Para el desarrollo de PetChain se ha seleccionado el modelo de licenciamiento CC BY-NC-SA 4.0¹⁷, el cual permite:
 - BY: Cualquier persona o institución es libre de usar la aplicación, con la obligación de citar al autor.
 - NC: No Commercial. No se permite el uso comercial de la aplicación.
 - SA: Share Alike: En caso de usar este mismo material, modificarlo y distribuirlo, es obligatorio realizarlo bajo el mismo tipo de licencia que el original.
- Freepick from <http://www.flaticon.com>: Todos los iconos utilizados para ilustrar la aplicación proceden de esta web, la cual permite la descarga gratuita de miles de imágenes en formato svg y png, con la única obligación de citar el origen de los

¹⁷ <https://creativecommons.org/licenses/by-nc-sa/4.0/>

mismos a través del enlace que se encuentra situado en esta zona de la pantalla y que como se ha indicado anteriormente está presente durante toda la navegación.

5.3 Alta de mascotas

Ilustración 33: Pantalla de Alta de mascotas

El proceso de alta de una mascota en el sistema se realiza a través de un formulario tradicional, en el que se realizan las validaciones de campos obligatorios, formato y reglas de negocio. Todos los errores relacionados con este punto, como la obligatoriedad de que la fecha de nacimiento sea anterior a la de implantación del chip, se muestran en ventanas emergentes reutilizables, tal y como se muestra en la siguiente imagen.



Ilustración 34: Ventana de aviso/error

Todas las validaciones de campos obligatorios son controladas, por HTML5, con lo que no difieren de cualquier otro modelo existente. Por último cuando el formulario está correctamente cumplimentado, al activar el alta el sistema siempre avisará al usuario para la confirmación de la acción, así como de su correcta finalización mediante sendos mensajes en ventanas emergentes.



Ilustración 35: Ventanas de confirmación y finalización.

5.4 Modificación de mascotas

Para el proceso de modificación, siguiendo criterios de reutilización para no repetir código y de usabilidad para mantener el mismo formato de formulario, se ha empleado el mismo fichero que el del Alta. Para ello se han aprovechado a fondo las funcionalidades de las vistas

y los componentes que ofrece Vue.js, pudiendo con ello ahorrarse tiempo de desarrollo y testing.

Como ya se ha comentado, el formulario es similar al del Alta pero con la particularidad de la existencia de campos deshabilitados que, obviamente, no pueden ser modificados, como la fecha de nacimiento, fecha de implantación del chip, raza, nombre, género, pelo o aptitud. Por otro lado aparece en este proceso la figura del veterinario modificador, que puede ser igual o distinto al identificador, fundamental este punto para la realización de transferencias en ETH, en caso de realizarse algún cambio sobre la mascota o el propietario.

Por último, al igual que en el proceso anterior, cualquier modificación realizada debe ser confirmada por el usuario.

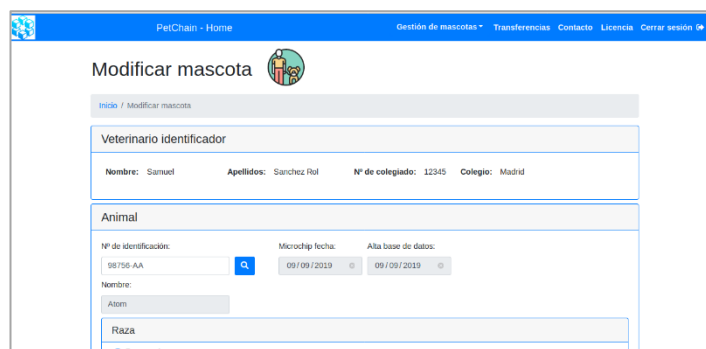


Ilustración 36: Pantalla de Modificación de mascotas

5.5 Baja de mascotas

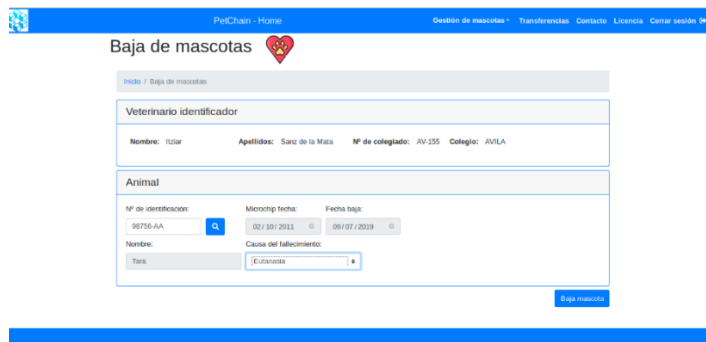


Ilustración 37: Pantalla de Baja de mascotas

Funcionalmente esta pantalla tiene un comportamiento muy sencillo e intuitivo, tan sólo solicita al usuario el identificador de la mascota a la que se desea dar de baja del sistema, e indicar el motivo del fallecimiento de la misma. El resto de los campos son de tipo informativo y se encuentran deshabilitados.

Al introducir el identificador, o por el campo asociado o a través de la ventana emergente que se muestra al seleccionar el elemento de búsqueda de la lupa, tal y como se realiza en la

Modificación de mascotas, el sistema devuelve la información del mismo, recuperando los datos del hash almacenado en el Smartcontract y yendo a recuperar el fichero a la IPFS.

Una vez recuperados los datos y pulsado el botón de eliminar, el sistema nos solicitará la confirmación sobre la eliminación, para dar de baja al animal del sistema.

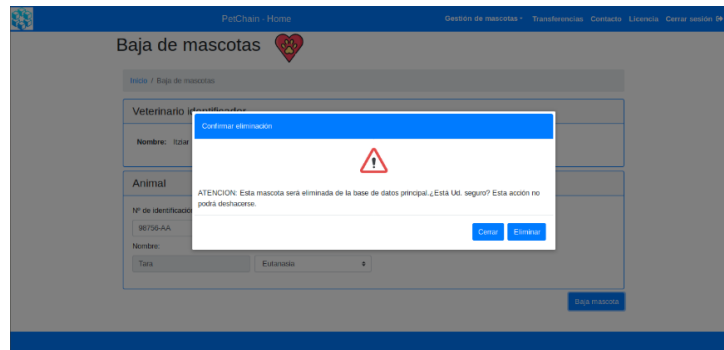


Ilustración 38: Confirmación de la Baja de una mascota.

Esta acción no eliminará los datos de manera permanente, dado que siempre quedarán almacenados en la IPFS y a través del hash es posible su recuperación.

5.6 Transferencias y saldo de ETHERS

En correspondencia con el FRQ-0008 es necesario visualizar en una pantalla las transferencias realizadas entre el veterinario que hubiese hecho alguna modificación a una mascota dada de alta por otro diferente. Es decir, tiene que existir una transferencia en ETH del veterinario modificador a favor del identificador.

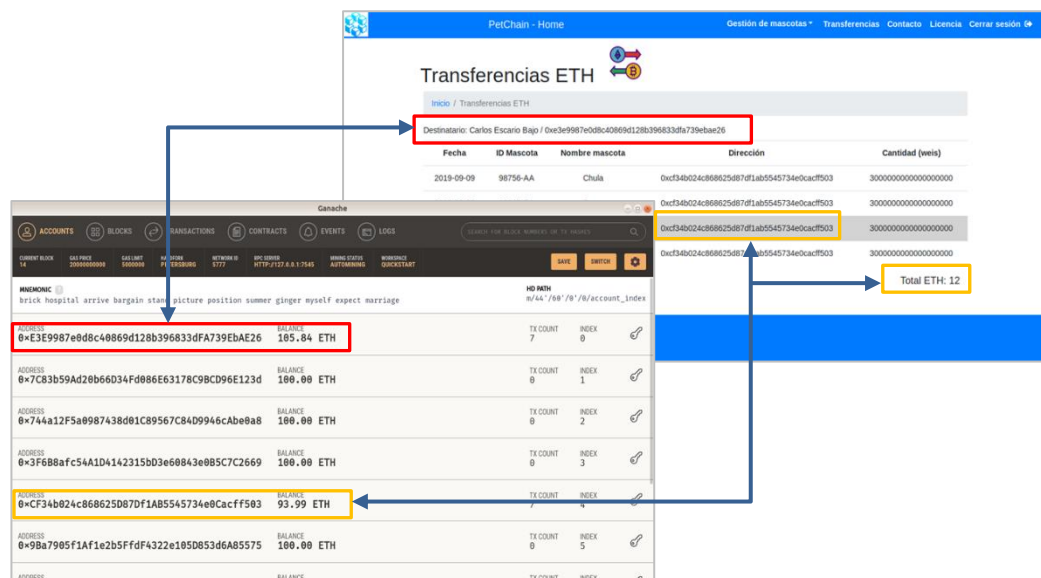


Ilustración 39: Tranferencias de ETH

En la imagen anterior se observa que por la realización de 4 modificaciones desde la cuenta modificadora (en amarillo) se ha realizado una transferencia de 3 ETH a la cuenta identificadora (en rojo), por cada una de ellas, su registro dentro la Blockchain y su saldo total a favor de ésta última.

6 Test de código.

El objetivo de esta sección es comprobar como el código desarrollado cumple con las expectativas, requisitos y criterios técnicos definidos. En concreto se van a realizar los siguientes test para la validación de la aplicación:

- Validación de código.
- Test de Smartcontracts.
- Validación de objetos JSON.
- Visualización en diferentes dispositivos.

6.1 Validación de código

En el capítulo 3 (PetChain. Especificación de requisitos) se hizo una enumeración de los requisitos necesarios para la implementación de Petchain, en concreto dentro de los requisitos no funcionales, se estableció la necesidad de la validación del código según los estándares del W3C (NFR-0008).

Para la realización de estas pruebas se han cargado los ficheros estáticos (*.html y *.css) en la página web <https://validator.w3.org/nu/#file> con el objeto de conocer el nivel de cumplimiento del código desarrollado.

6.1.1 Validación inicial (*.html)

Basándonos en las pruebas realizadas, los ficheros *.html de Petchain arrojan los siguientes resultados:

| Pruebas de validación de código (*.html) | | |
|--|--|---|
| Login.html | • OK | ✓ |
| Contacto.html | • Errores: 6 • Warnings: 6 | ✗ |
| Addpet.html | • Errores: 9 • Warnings: 6 | ✗ |
| Deletepet.html | • Errores: 9 • Warnings: 4 | ✗ |
| Updatepet.html | • Errores: 12 • Warnings: 6 | ✗ |
| Licencia.html | • Errores: 2 • Warnings: 6 | ✗ |
| index.html | • Errores: 6 | ✗ |

Tabla 55: Resultados de la validación inicial de los ficheros html

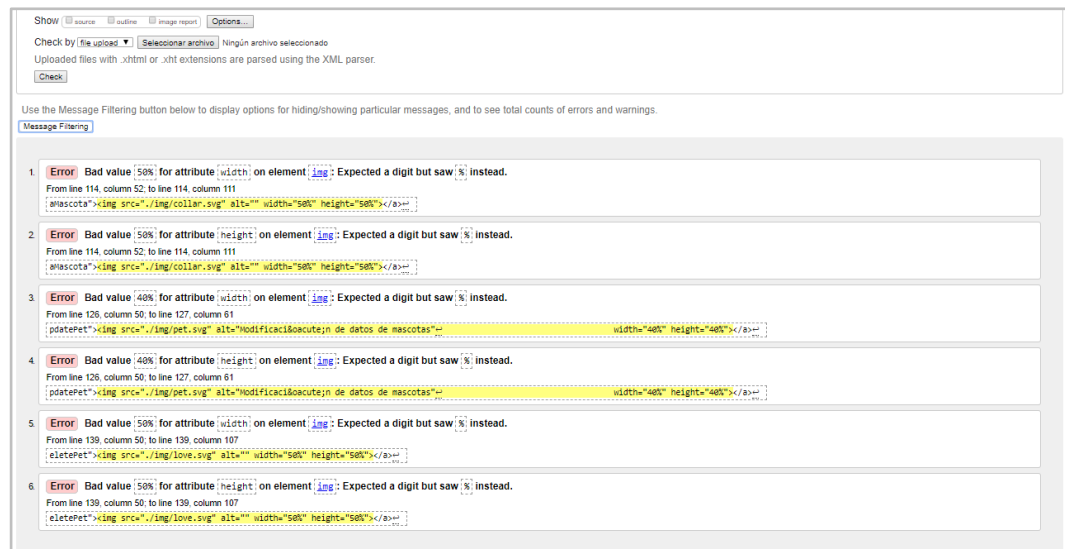


Ilustración 40: : Resultados W3C de la validación de index.html (Fuente: <https://validator.w3.org/>)

Los errores detectados en los ficheros html corresponden a:

- El uso deprecado o desaconsejado de atributos definidos en línea sin un valor numérico explícito, es decir usando “px” o “%” dentro de las etiquetas. Por ejemplo como se muestra en la imagen anterior, para la definición de atributos alto y ancho se aconseja el uso de estilos css (inline o en una clase) o bien la definición de un valor numérico sin modificadores en píxeles o porcentaje.
- En la línea del punto anterior, el uso de estos mismos atributos dentro de la definición del iframe, para determinar su alto y ancho.
- Definición de etiquetas `<option value="">`, en donde el atributo “value” no puede contener un valor vacío aunque sea el elemento inicial que actúa a modo de etiqueta descriptiva. Se aconseja siempre la definición de un valor dentro de cualquier atributo html.

Por otro lado, el validador muestra advertencias o *warnings* sobre elementos que, a pesar de no considerarlos correctos, no invalidan el código en su totalidad. Como ejemplo de este tipo de avisos tenemos:

- Los generados por el uso innecesario del atributo `type="text/javascript"` dentro de las etiquetas `<script type="text/javascript">`. El problema de este tipo de atributos, es que son autogenerados por el framework Vue.js en cada proceso de compilación y no es posible eliminarlo, con lo que se ha optado por no cambiar nada en la configuración de Vue y asumir este tipo de mensajes.
- Advertencias por el uso del atributo “date” dentro de las etiquetas “input” que se usan para la definición de los campos de los formularios: `<input type="date" .../>`. En este caso el validador las desaconseja debido a que no están soportadas

todavía por el 100% de los navegadores. En esta ocasión, también se asume esta advertencia debido a que se quiere mostrar en pantalla un control de tipo calendario que sea lo más sencillo posible, en lugar de usar algún plugin de terceros que necesitase la carga de una librería externa.

Tras la revisión del código y realizadas las correcciones que propone el W3C, el resultado es el siguiente:

| Pruebas de validación de código (*.html) | | |
|--|----------------------|---|
| Login.html | • OK | ✓ |
| Contacto.html | • Warnings: 1 | ✓ |
| Addpet.html | • Warnings: 4 | ✓ |
| Deletepet.html | • Warnings: 2 | ✓ |
| Updatepet.html | • Warnings: 5 | ✓ |
| Licencia.html | • Warnings: 5 | ✓ |
| index.html | • OK | ✓ |

Tabla 56: Resultados del código html validado

Ilustración 41: Validación W3C index.html (Fuente: <https://validator.w3.org/>)

6.1.2 Validación inicial (*.css)

Los resultados de las pruebas realizadas a estos ficheros se muestran a continuación en el cuadro resumen.



ATENCIÓN: No es objeto del presente análisis el realizar pruebas de validación a ficheros *.css pertenecientes al framework Bootstrap 4.


| Pruebas de validación de código (*.css) | | |
|---|---|---|
| main.css | <ul style="list-style-type: none"> Errores: 2 |  |

Tabla 57: Resultados de la validación de main.css

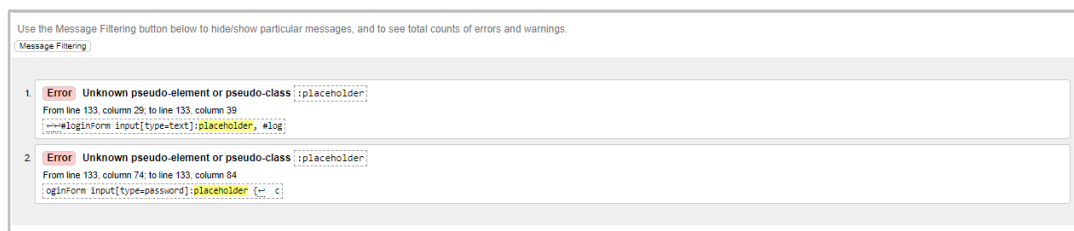


Ilustración 42: Resultados W3C de main.css (Fuente: <https://validator.w3.org/>)

La pseudoclase “:placeholder”, empleada como modificador del comportamiento dentro de un css, no está definida en la especificación CSS3 del W3C. Sin embargo si está en fase de estudio para incorporarla a la definición que vendrá de CSS4, que en la actualidad está desarrollando el CSSWG¹⁸, por este motivo, se permite incorporarla:

- Con el prefijo –ms, –moz o -webkit: para restringir su uso a navegadores de la familia Microsoft, Mozilla o Chrome, indicándole al validador que es una característica propia de ellos, y que por lo tanto la de por válida.
- Con el prefijo ‘::’, para definirla como un pseudoelemento en lugar de pseudoclase.

Modificando el estilo con la primera opción, obtenemos el resultado esperado.


| Pruebas de validación de código (*.css) | | |
|---|---|---|
| main.css | <ul style="list-style-type: none"> OK |  |

Tabla 58: Resultado de la validación de main.css

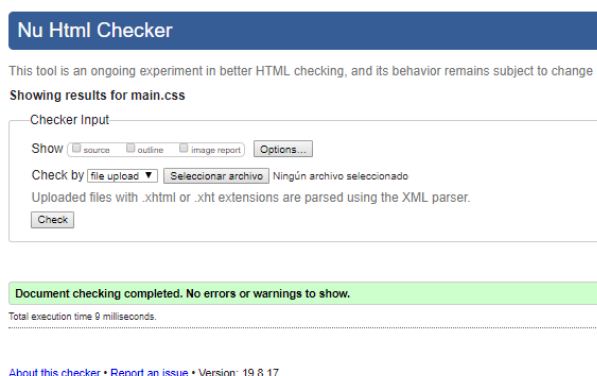


Ilustración 43: Resultado final del W3C de la validación de main.css (Fuente: <https://validator.w3.org/>)

¹⁸ <https://wiki.csswg.org/>

6.2 Test Smartcontracts

Los test de los Smartcontracts desplegados en la Blockchain se van a realizar sobre aquellos métodos que devuelven datos y observando que éstos se ajustan al resultado esperado en base a los parámetros de entrada suministrados.

La ejecución de los test se realizará sobre la herramienta existente en la red, y destinada para este propósito, que se encuentra disponible de manera gratuita en la url <https://remix.ethereum.org>. Aquí se encuentra un IDE y un entorno de ejecución tipo SANDBOX, que permite diseñar, compilar y ejecutar Smartcontract sobre múltiples versiones del compilador de Solidity y entornos de ejecución como VM Javascript o Web3.js.

Para nuestras pruebas, tal y como se muestra en la cabecera de todos los ficheros Solidity, la versión del compilador empleada es la 0.4.24 y el entorno de ejecución es la máquina virtual de Javascript.

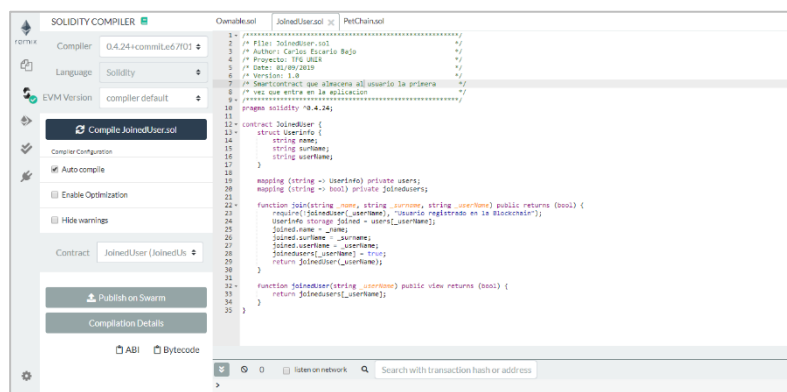


Ilustración 44: remix.ethereum.org IDE (Fuente: <https://remix.ethereum.org>)

6.2.1 Test JoinedUser.sol

En este fichero de Solidity tenemos el Smartcontract *JoinedUser*, que es el encargado de registrar a los usuarios en la Blockchain en ejecución. Para ello cuenta con los métodos siguientes:

| JoinedUser.sol | | |
|----------------|--|----------------|
| Método | Recibe | Valor esperado |
| join | _name: String _surname: String _userName: String | True/false |
| joinedUser | _userName: String | True/false |

Tabla 59: Descripción de JoinedUser.sol

Ejecución del test

| JoinedUser.sol | | | |
|----------------|-----------|---|----|
| Método | Ejecución | | |
| join | I | { "string _name": "Carlos", "string _surname": "Escario Bajo", "string _userName": "carlosecariobajo@gmail.com" } | OK |
| | O | { "0": "bool: true" } | |
| joinedUser | I | { "string _userName": "carlosecariobajo@gmail.com" } | OK |
| | O | { "0": "bool: true" } | |

Tabla 60: Ejecución de test sobre JoinedUser.sol

6.2.2 Test PetChain.sol

| PetChain.sol | | |
|--------------|--------------------------------------|----------------|
| Método | Recibe | Valor esperado |
| setPet | _petId: String _data_hash: String | True |
| getPet | _petId: String | String, String |

Tabla 61: Descripción de PetChain.sol

Ejecución del test

| PetChain.sol | | | |
|--------------|-----------|--|----|
| Método | Ejecución | | |
| setPet | I | { "string _pet_id": "12345-AA", "string _data_hash": "QmcK1hAc9sYTHJZcr68HxdHLCMPFgSn9QMSSar7j85TFg5" } | OK |
| | O | { "0": "bool: true" } | |
| getPet | I | { "string _pet_id": "12345-AA" } | OK |
| | O | { "0": "string: 12345-AA", "1": "string: QmcK1hAc9sYTHJZcr68HxdHLCMPFgSn9QMSSar7j85TFg5" } | |

Tabla 62: Ejecución de test sobre PetChain.sol

6.2.3 Test registerTX

| registerTX .sol | | |
|-----------------|---|----------------|
| Método | Recibe | Valor esperado |
| setTX | _toAddr: Address _fromAddr: Address _txHash: String | True |
| getTX | _toAddr: Address | String, String |

Tabla 63: Descripción de registerTX.sol

Ejecución del test

| registerTX.sol | | | |
|----------------|-----------|--|----|
| Método | Ejecución | | |
| setTX | I | { "address _toAddr": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c", "address _fromAddr": "0x14723A09ACff6D2A60DcdF7aA4AFf308FDDC160C", "string _txhash": "0x00" } | OK |
| | O | { "0": "bool: true" } | |
| getTX | I | { "address _toAddr": "0xCA35b7d915458EF540aDe6068dFe2F44E8fa733c" } | OK |
| | O | { "0": "string: 0x00" } | |

Tabla 64: Ejecución de test sobre registerTX.sol

6.2.4 Resultados de los test

Para ambas ejecuciones los resultados que se muestran, son los ofrecidos por la consola del SANDBOX de Remix, y en los dos casos la salida obtenida coincide con el valor esperado. Por otro lado, la gran ventaja que nos ofrece la ejecución de los Smartcontract en la Blockchain es que cualquier valor recibido por el método y que no coincida con el tipo esperado, provoca un *rollback* de la transacción, con lo que con este comportamiento nos aseguramos de que la ejecución siempre será exitosa.

6.3 Validación de objetos JSON

Los objetos JSON empleados en PetChain deben de comportarse como elementos con una notación estándar estricta para favorecer su uso y su posible mantenimiento. Para validar que su estructura es correcta, se ha empleado una de las herramientas online más reputadas y que se ofrece desde la web <https://jsonformatter.org/>.

La validación se ha realizado sobre tres de los ficheros existentes en /src/data/, dando como resultado:

| Pruebas de validación de objetos JSON (*.json) | | |
|--|------|---|
| users.json | • OK | ✓ |
| países_ue.json | • OK | ✓ |
| provincias_es.json | • OK | ✓ |

Tabla 65: Resultados validación de ficheros JSON

6.4 Visualización en diferentes dispositivos

En respuesta al NFR-0001¹⁹, en el que se establece la necesidad no funcional de crear un diseño reponsive, es decir un desarrollo web que se visualice y se pueda interactuar con él, independientemente del dispositivo con el que se acceda, Smartphone, Tablet, portátil o PC, se van a testear PetChain para comprobar su funcionamiento en diferentes tamaños de pantalla.

Se van a probar tres tipos de resoluciones:

- Hasta un ancho máximo de 575,98 píxeles: que corresponden a los tamaños de dispositivos Smartphone en horizontal y vertical.
- Entre 576 y 1199,98 píxeles: que corresponden a tamaños de Tablet en horizontal y vertical o dispositivos convertibles tipo Microsoft® Surface.
- Por último para dispositivos con un ancho mínimo de 1200 píxeles, que corresponderían a PC y portátiles de uso común.

6.4.1 Realización de pruebas

Las pruebas se realizan sobre la visualización de los siguientes elementos:

- Para resoluciones inferiores en ancho a 768px (Smartphones y Tablets en vertical), será necesario que:
 - Las opciones del menú superior desaparezcan y se visualice en su lugar en la esquina superior derecha de la pantalla un icono con las tres líneas representativas del menú colapsado.
 - El menú debería desplegarse de arriba hacia abajo mostrando las opciones en vertical.
 - El logo de la aplicación situado en a la izquierda del menú deberá desaparecer.
 - Las imágenes de la página principal deberán reducir su tamaño hasta un 25% del ancho de la pantalla situándose en una única columna.

¹⁹ Sección 3.8.3. Requisitos no funcionales

- Para resoluciones entre 576 y 1199,98 píxeles (Tablet y PC):
 - Reducción de las imágenes hasta un 50% de su tamaño.
- Resto de resoluciones sin cambios.

Al haber empleado para el desarrollo de la aplicación Bootstrap 4 y habiendo utilizado para la maquetación en formato grid las reglas css que establece para dispositivos pequeños, medianos y grandes, se cede el control de la disposición de los campos de los formularios al framework, con lo que no se van a realizar test a esta clase de elementos.

Para su realización se va a emplear la utilidad que posee el navegador Firefox 67.0, que se activa dentro de las opciones del desarrollador. Asimismo, también para comprobar los resultados obtenidos con el navegador se ha utilizado la web <http://responsivetesttool.com/>, que con una interfaz intuitiva, permite comprobar el funcionamiento en una amplia gama de resoluciones.

6.4.2 Test Smartphone

- Desaparición del menú superior: como se observa en la imagen siguiente, se produce el efecto buscado con lo que se consigue para una simulación de iPhone 7 (414x736) en orientación vertical y para un LG Optimus L70 en horizontal.

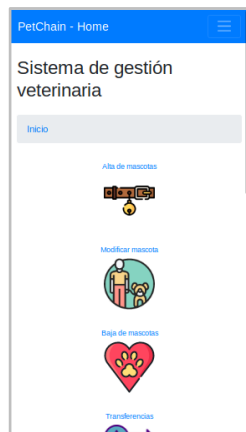


Ilustración 46: Simulación de Iphone7

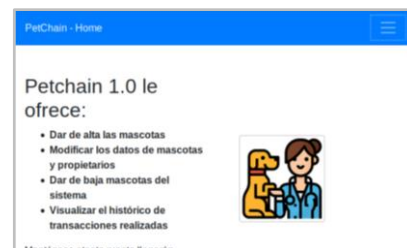


Ilustración 45: Simulación de LG Optimus L70

- Despliegue vertical de las opciones de menú: Se testea este funcionamiento en una simulación de iPad (768x1024) en orientación vertical y de un iPhone7 (736x414) en horizontal

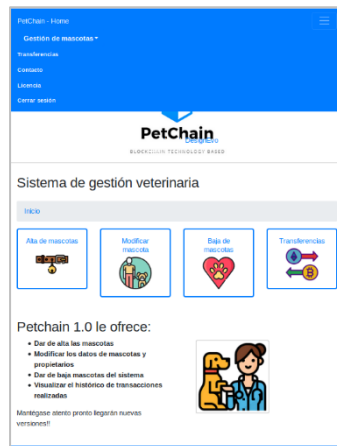


Ilustración 48: Simulación de iPad

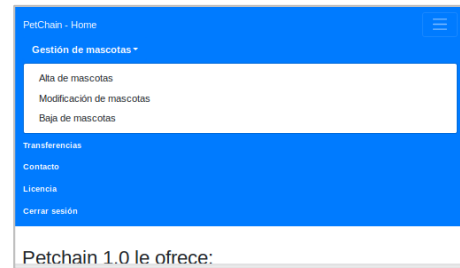


Ilustración 47: Simulación iPhone7 en horizontal

- Para todas las simulaciones anteriores se observa como se produce la desaparición del logo que se sitúa en la esquina superior izquierda.
- Asimismo, se observa en la simulación del iPhone7 la distribución vertical de las imágenes de la pantalla principal para este tipo de resoluciones.

6.4.3 Test Tablet

- Las imágenes, tal y como se muestra en la simulación anterior del iPad, se produce la disminución del tamaño de las imágenes distribuyéndose uniformemente a lo ancho de la pantalla.

7 Conclusiones y trabajo futuro.

En el presente capítulo se pretende hacer un análisis del Proyecto en relación a los objetivos planteados en el punto 1.3 (Objetivos), y su relación con el reto definido en el punto 1.2 (Motivación). A continuación, se quieren plantear las evoluciones que serían deseables para PetChain, de cara a un despliegue en una Blockchain pública y con vistas a un hipotético funcionamiento en producción.

7.1 Análisis sobre objetivos establecidos

Al inicio del presente trabajo se han planteado una serie de objetivos, uno de carácter general y otra serie de ellos como específicos a los que se ha querido dar respuesta con el presente desarrollo.

Como objetivo general se había establecido el desarrollar una DApp que ejecutándose en una Blockchain permitiese al conjunto de veterinarios identificar a las mascotas, independientemente de la Comunidad Autónoma en donde desarrollen su labor, pudiendo acceder tanto a los datos de éstas como a los de sus propietarios. Tras el desarrollo de la misma y analizando su funcionamiento se puede concluir que:

- El entorno de ejecución, la web, hace idóneo el acceso a la aplicación en cualquier lugar. Esta característica, junto con el diseño responsive permitiría el poder trabajar con ella o en una clínica con un PC o bien con cualquier dispositivo móvil Tablet o Smartphone.
- El modelo del Backend es transparente para el usuario, pues es independiente del frontal, y el hecho del alojamiento de la información en la IPFS no debe ser de importancia para él.
- Se evita el diseño de middlewares (a veces costosos en tiempo y dinero) para la interconexión de los diferentes sistemas existentes en la actualidad.

Dentro del apartado de objetivos específicos, como primer objetivo se planteó el reto de demostrar la viabilidad de la Blockchain para la realización de proyectos de gestión. Se puede comprobar que esto es ya una realidad tanto por el propio funcionamiento de PetChain como por la multitud de DApps que han surgido en estos últimos años. Como muestra de ello, el gráfico siguiente refleja la evolución de su desarrollo durante el periodo Abril-2015/Agosto-2019.

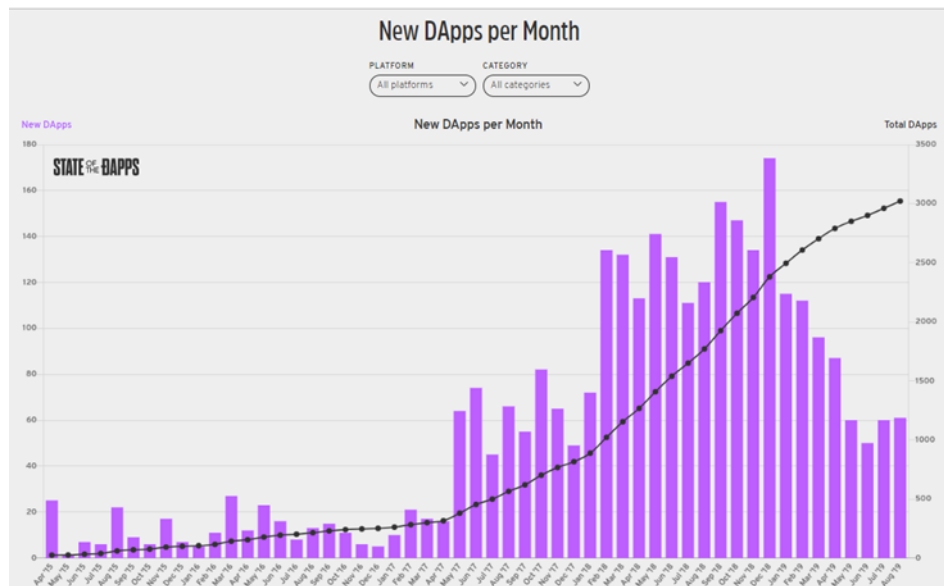


Ilustración 49: Evolución del desarrollo de DApps (Fuente: <https://www.stateofthedapps.com/stats>)

En el segundo punto de estos objetivos pretendía crear una web atractiva, bajo un criterio de “menos es más” y para ello se ha buscado un diseño sencillo y que no provocase “fatiga visual” en el usuario. Que fuese usable, es decir que no hiciese pensar al usuario donde están las opciones y que hay que hacer en todo momento, aportando para ello imágenes claras, mensajes informativos, de advertencia y error para dar al usuario sensación de tranquilidad y solidez. En este punto las pruebas realizadas por la Dirección facultativa del proyecto han dado su completa satisfacción a este objetivo.

El desarrollo de un proyecto con software de coste cero está absolutamente conseguido, pues todas las herramientas utilizadas, tanto frameworks como entornos de desarrollo son de libre uso y su descarga es gratuita a través de Internet.

El objetivo de no ser un experto en criptomoneda es un hecho, pues el desarrollo dentro de este modelo no implica conocimientos financieros de este nivel. Señalar en este punto que si es necesario que el desarrollador conozca a nivel técnico conceptos como “transacción”, “gas” o “gas limit”, que si son imprescindibles para el desarrollo de los Smartcontracts y su ejecución de cara a una implantación en entornos productivos.

7.2 Trabajo futuro

Para una adopción del proyecto en un entorno productivo se hacen necesarios diversos trabajos de carácter obligatorio, los cuales se exponen a continuación:

- Implementación de un sistema de acceso seguro. Este apartado se conseguiría a través de un despliegue en una Blockchain privadas, tipo HyperLedger²⁰, que implicasen el acceso a las mismas a través de un sistema de Directorio Activo.
- Implementación de nuevas funcionalidades:
 - Alta, baja y modificación de profesionales veterinarios.
 - Solicitud de compra de chips a los Colegios Veterinarios territoriales.
 - Sistema de mensajería de imágenes instantáneo (RX y fotos del animal), para la compartición de expediente y diagnósticos veterinarios, bajo un sistema de recompensa en ETH para el veterinario origen.
 - Consenso sobre la cantidad de ETH a abonar entre veterinarios
 - Registro de vacunaciones.
 - Registro de animales perdidos.
- Registro del fichero de datos en la AEPD, para el correcto manejo de los datos de carácter personal de los propietarios y del personal veterinario
- Creación de un fichero de propiedades para la adecuación de la aplicación a los diferentes idiomas autonómicos.
- Adecuación de la interfaz gráfica a un estándar de accesibilidad mínimo AA.

²⁰ <https://hyperledger.org>

8 Bibliografía

- Bassagañas Galisteo, Jordi (2013). *MVC CodeIgniter para simpáticos newbies I (El gran curso de desarrollo de apps MVC nº 1)* España. Edición de Kindle. Amazon Media EU S.à r.l.
- Benet, Juan (2014), *IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)* <https://github.com/ipfs/ipfs/blob/master/papers/ipfs-cap2pfs/ipfs-p2p-file-system.pdf>
- Cash, Mercury (2018). *Aplicaciones de la tecnología blockchain y su relación con los sistemas descentralizados*. <https://blog.mercury.cash>
- Duran, Amador (2000). *Un Entorno Metodológico de Ingeniería de requisitos para Sistemas de la Información*. España: Universidad de Sevilla.
- Gupta, Manav (2018). *Blockchain*. E.E.U.U: IBM
- Modi, Ritesh (2018) *Solidity Programming Essentials. A beginner's guide to build Smartcontracts for Ethereum and blockchain*. UK: Packt Publishing.
- Somerville, Ian (2009). *Ingeniería de Software*. México: Pearson educación.
- Venter, Ryan (2016). *The Modern Ethereum*. UK: Amazon Media EU S.à r.l.

9 Glosario de términos

A

AEPD
 Agencia Española de Protección
 de Datos.....86

API
 Application Program Interface....1, 14, 59
 62

B

C

CCAA
 Comunidades Autónomas.....10

CDN
 Content Delivery Network.....63

CSSWG
 CSS Working Group.....77

CSS3
 Cascade Style Sheet ver.3.....26, 77

D

DNI
 Documento nacional de
 Identidad.....10, 25

E

ECMA
 European Computer Manufactures
 Association.....26

ETH
 Ethers.....7, 25
 30, 33, 37, 38, 46, 70, 71, 72, 85

F

FRQ
 Functional Requirement.....32, 35, 36,
 37, 38, 39, 45, 66, 67, 72

G

H

HTML5
 Hiper Text Markup Language.....26,
 67, 70

http
 hyper text transfer protocol.....62.

I

IDE
 Integrated Development
 Environment.....12, 77

IPFS
 Inter Planetary File System.....1, 2, 3,
 5, 12, 14, 20, 23, 25, 27, 46, 48, 51, 55, 56,
 72, 84

J

JSON
 Javascript Object Notation.....8,
 17, 23, 25, 27, 41, 43, 49, 50, 51, 54, 55,
 60, 61, 74, 80, 81

JSX
 Javascript XML.....59

K, L

M

MVC

| | | | |
|---|------------------------|---|------------------------|
| Model View Controller..... | 5, 27, 49 | Remote Procedure Call..... | 61 |
| Mac OS | | S | |
| Macintosh Operating System..... | 60, 61, 64 | SIACAM | |
| N | | Sistema de Identificación Animal de Castilla la Mancha..... | 10 |
| NFR | | SIACYL | |
| No Functional Requirement..... | 40, 41, 42, 43, 45, 74 | Sistema de Identificación Animal de Castilla y León..... | 10 |
| NoSQL | | SIAMEL | |
| No Structured Query Language..... | 25, 26 | Sistema de Identificación Animal de la Ciudad de Melilla..... | 10 |
| npm | | SIAMU | |
| Node package manager..... | 57, 59, 61 | Sistema de Identificación Animal de la Región de Murcia..... | 10 |
| O | | SPA | |
| P | | Single Page Application..... | 58 |
| P2P | | T | |
| Peer to peer..... | 14, 21, 86 | TFG | |
| PC | | Trabajo de fin de grado..... | 22 |
| Personal Computer..... | 25, 29, 61, 80, 81, 83 | U | |
| PHP | | UNIR | |
| HyperText Preprocessor..... | 64 | Universidad Internacional de la Rioja..... | 1 |
| Q | | URL | |
| R | | Uniform Resource Locator..... | 54, 56, 58, 59, 62, 77 |
| REM | | V | |
| Requirements Management..... | 63 | VM | |
| RIACA | | Virtual Machine..... | 77 |
| Registro de Animales de Compañía de Aragón..... | 10 | W | |
| RPC | | W3C | |

World Wide Web Consortium.....5, 6,
42, 73, 74, 75, 76, 88

X

XML

Extensible Markup Language.....87

Y, Z

10 Referencias

Iconos y elementos gráficos: Icon made by [Freepik](#) from www.flaticon.com