

|  |
| --- |
| **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/tfg_project>

**Trabajo Fin de Grado**

**Presentado por:** Escario Bajo, Carlos

**Director/a:** Martinez Muñoz, Miriam.

Ciudad: San Lorenzo de El Escorial (Madrid)

Fecha: 05/06/2019

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, Ganaché, Truffle, 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 inteded 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 of 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, token, node, IPFS.

**Indice**

[Resumen 1](#_Toc10417941)

[Summary 1](#_Toc10417942)

[Índice de ilustraciones 5](#_Toc10417943)

[Índice de Tablas 6](#_Toc10417944)

[1. Introducción 8](#_Toc10417945)

[1.1. Presentación del problema 8](#_Toc10417946)

[1.2. La solución. 9](#_Toc10417947)

[1.1.1. API de servicios web. 10](#_Toc10417948)

[1.1.2. Blockchain 11](#_Toc10417949)

[2. Estado del arte. 12](#_Toc10417950)

[2.1. La Blockchain 12](#_Toc10417952)

[2.1.1. Características 12](#_Toc10417956)

[2.1.2. ¿Qué es un bloque? 12](#_Toc10417957)

[2.1.3. ¿Dónde están los bloques? 14](#_Toc10417958)

[2.1.4. Criptografía de clave pública 14](#_Toc10417959)

[3.1. IPFS 14](#_Toc10417960)

[3. Petchain. Especificación de requisitos. 15](#_Toc10417961)

[5.1. Propósito del documento 15](#_Toc10417964)

[5.2. Alcance del documento 15](#_Toc10417965)

[5.3. Participantes en el proyecto 15](#_Toc10417966)

[5.3.1. Presentación de participantes 15](#_Toc10417973)

[5.4. Organizaciones involucradas 16](#_Toc10417974)

[5.5. Descripción del sistema 17](#_Toc10417975)

[5.5.1. Perspectiva del producto 17](#_Toc10417978)

[5.5.2. Funciones del producto 17](#_Toc10417979)

[5.5.3. Características del usuario 18](#_Toc10417980)

[5.5.4. Restricciones 18](#_Toc10417981)

[5.6. Descripción técnica del sistema 18](#_Toc10417982)

[5.6.1. Front-End 18](#_Toc10417984)

[5.6.2. Seguridad 19](#_Toc10417985)

[5.6.3. Back-End 19](#_Toc10417986)

[5.6.4. Entorno de ejecución 19](#_Toc10417987)

[5.7. Objetivos del sistema 20](#_Toc10417988)

[5.8. Catálogo de requisitos del sistema. 24](#_Toc10417989)

[5.8.1. Requisitos de información 24](#_Toc10417992)

[5.8.2. Requisitos funcionales 27](#_Toc10417993)

[5.8.3. Requisitos no funcionales 32](#_Toc10417994)

[5.9. Matriz de rastreabilidad 36](#_Toc10417995)

[5.9.1. Objetivos del sist. – Requisitos funcionales / Requisitos no funcionales 36](#_Toc10417997)

[5.10. Casos de uso 37](#_Toc10417998)

[5.10.1. Gestión de mascotas 37](#_Toc10418000)

[5.10.2. Gestión de propietarios 37](#_Toc10418001)

[5.10.3. Consulta de saldo 38](#_Toc10418002)

[4. Metodología y Plan de trabajo 39](#_Toc10418003)

[5.11. Seguimiento del proyecto 39](#_Toc10418004)

[5. Petchain. Descripción técnica. 40](#_Toc10418005)

[6.1. NodeJS 41](#_Toc10418007)

[6.1.1. ExpressJS 41](#_Toc10418010)

[6.1.2. body-parser 41](#_Toc10418011)

[6.1.3. jsonwebtoken 42](#_Toc10418012)

[6.1.4. Generación del proyecto 42](#_Toc10418013)

[6.2. Truffle 43](#_Toc10418014)

[6.2.1. Instalación de Truffle 43](#_Toc10418017)

[6.2.2. Generación del proyecto 43](#_Toc10418018)

[6.3. Ganache 44](#_Toc10418019)

[6.3.1. Instalación de Ganache 44](#_Toc10418021)

[6.4. Web3.js 45](#_Toc10418022)

[6.4.1. Instalación de Web3.js 45](#_Toc10418024)

[6.5. JQuery 45](#_Toc10418025)

[6.6. Bootstrap 46](#_Toc10418026)

[6.7. Herramientas utilizadas 47](#_Toc10418027)

[6.8. Patrón de diseño. 47](#_Toc10418028)

[6.4.2. Modelo 47](#_Toc10418029)

[6.4.3. Vista 48](#_Toc10418030)

[6.4.4. Controlador 48](#_Toc10418031)

[6. Petchain. Descripción funcional 50](#_Toc10418032)

[7. Evaluación 50](#_Toc10418033)

[8. ¿Qué queda por hacer? Trabajo futuro. 50](#_Toc10418034)

[9. Bibliografía 51](#_Toc10418035)

[10. Glosario de términos 52](#_Toc10418036)

[11. Referencias 54](#_Toc10418037)

Índice de ilustraciones

[Ilustración 1: Petchain logo 1](#_Toc10417758)

[Ilustración 2: Representación de la cadena de bloques. 12](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417759)

[Ilustración 3: Ejemplo de bloque genesis.json 12](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417760)

[Ilustración 4: Bloque Genesis 13](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417761)

[Ilustración 5: Gestión de mascotas 36](#_Toc10417762)

[Ilustración 6: Gestión de propietarios 37](#_Toc10417763)

[Ilustración 7: Consulta del saldo en Ethers 37](#_Toc10417764)

[Ilustración 8: Tablero principal de Petchain en Trello 38](#_Toc10417765)

[Ilustración 9: Esquema gráfico de Petchain (Fuente: diseño propio) 39](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417766)

[Ilustración 10: Comando instalación de ExpressJS 40](#_Toc10417767)

[Ilustración 11: Comando de instalación de body-parser 40](#_Toc10417768)

[Ilustración 12: Comando de instalación de jsonwebtoken 41](#_Toc10417769)

[Ilustración 13: Comandos de creación directorio del proyecto 41](#_Toc10417770)

[Ilustración 14: Inicialización del proyecto NodeJS 41](#_Toc10417771)

[Ilustración 15: Estructura del fichero package.json 41](#_Toc10417772)

[Ilustración 16: : Comando de instalación de Truffle 42](#_Toc10417773)

[Ilustración 17: Comando de inicialización de un proyecto Truffle 42](#_Toc10417774)

[Ilustración 18: Web de Truffle framework. 43](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417775)

[Ilustración 19: Estructura de vendor/jquery 45](#_Toc10417776)

[Ilustración 20: Estructura de vendor/bootstrap 45](#_Toc10417777)

[Ilustración 21: Esquema clásico del patrón MVC. Fuente: Diseño propio 46](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc10417778)

[Ilustración 22: Estructura de los ficheros del Modelo 47](#_Toc10417779)

[Ilustración 23: Estructura de ficheros de la Vista 47](#_Toc10417780)

Índice de Tablas

[Tabla 1: Dirección del proyecto 15](#_Toc10417788)

[Tabla 2: Analista y desarrollador del proyecto. 15](#_Toc10417789)

[Tabla 3: Colaborador del proyecto 15](#_Toc10417790)

[Tabla 4: Dirección facultativa y Veterinaria 15](#_Toc10417791)

[Tabla 5: Organización 1 16](#_Toc10417792)

[Tabla 6: Organización 2 16](#_Toc10417793)

[Tabla 7: Objetivo de Alta de mascotas 19](#_Toc10417794)

[Tabla 8: Objetivo Alta de dueños. 19](#_Toc10417795)

[Tabla 9: Objetivo de Modificación de datos 20](#_Toc10417796)

[Tabla 10: Obetivo de Baja de mascotas 20](#_Toc10417797)

[Tabla 11: Objetivo de Consulta de saldo de ETH 21](#_Toc10417798)

[Tabla 12: Objetivo de control de acceso al sistema 21](#_Toc10417799)

[Tabla 13: Objetivo de visulaización en múltiples disposivos. 21](#_Toc10417800)

[Tabla 14: Objetivo de traspaso de ETH 22](#_Toc10417801)

[Tabla 15: Objetivo de la navegación de la aplicación 22](#_Toc10417802)

[Tabla 16: Objetivo de Gestión de sesiones 23](#_Toc10417803)

[Tabla 17: Requisito de Informacion del veterinario 23](#_Toc10417804)

[Tabla 18: Requisitos de información de mascotas 24](#_Toc10417805)

[Tabla 19: Requisitos de información de propietarios. 25](#_Toc10417806)

[Tabla 20: Requisitos de información del saldo en ETH 26](#_Toc10417807)

[Tabla 21: Requisito funcional de acceso al sistema 26](#_Toc10417808)

[Tabla 22: rquisito funcional de identificación de verinarios. 27](#_Toc10417809)

[Tabla 23: Requisito funcional de traspaso de weis 27](#_Toc10417810)

[Tabla 24: Requisito funcional de navegación en el sistema. 28](#_Toc10417811)

[Tabla 25: Requisito funcional de modificación de datos de una mascota 28](#_Toc10417812)

[Tabla 26: Requisito funcional para la baja de una mascota 29](#_Toc10417813)

[Tabla 27: Requisito funcional para la visualización de weis disponibles 29](#_Toc10417814)

[Tabla 28: Requisito funcional para el alta de una mascota. 30](#_Toc10417815)

[Tabla 29: Requisito funcional para el alta de un propietario. 30](#_Toc10417816)

[Tabla 30: Requisito funcional para la modificación de un propietario. 31](#_Toc10417817)

[Tabla 31: Requisito no funcional de visualización de Petchain. 31](#_Toc10417818)

[Tabla 32: Requisito no funcional del pago en weis 32](#_Toc10417819)

[Tabla 33: Requisito no funcional del tiempo de sesión. 32](#_Toc10417820)

[Tabla 34: Requisito no funcional de usuarios autorizados 33](#_Toc10417821)

[Tabla 35: Requisito no funcional del nº de cuentas en Ganache 33](#_Toc10417822)

[Tabla 36: Requisito no funcional para la autenticación JWT 33](#_Toc10417823)

[Tabla 37: Requisito no funcional de campos obligatorios 34](#_Toc10417824)

[Tabla 38. Matriz de dependencias de requisitos 35](#_Toc10417825)

[Tabla 39: Relación Arquitectura - tecnología 46](#_Toc10417826)

# 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.

Se quiere 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.

## 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 labores 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 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, y en cuestión de horas, por razones personales, de trabajo o de ocio.

En España, según datos de la web de profesionales del sector veterinario Petshopmagazine[[1]](#footnote-1), 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**”. Este 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 SIACYL[[2]](#footnote-2), 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 casos como los que ocurren con asiduidad, en alguno de los servicios veterinarios de la provincia de Ávila que se encuentran 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 estos casos, es necesario realizar una llamada telefónica a los servicios veterinarios de Madrid y en donde sólo existe atención telefónica en horario laboral y de Lunes a Viernes, con lo que se dan circunstancias en las que un animal que se extravía, y el si sábado por la mañana se intenta localizar al dueño, esto no sea posible hasta el Lunes siguiente, y sólo mediante una llamada de telefónica.

## La solución.

Entonces, ¿qué se puede hacer? Evidentemente, conectar los sistemas de cada una de las 17 comunidades autónomas. ¿Cómo? La respuesta a esta pregunta es que de múltiples maneras, pero en este punto nos vamos a centrar en lo más actual. Podremos hacerlo a través de una **API de servicios web** o a través de la creación de una **Blockchain**.

|  |
| --- |
| Es necesario resaltar la idea de que el objeto del presente trabajo no es reflejar las ventajas de un sistema sobre otro, si no el mostrar la existencia de un nuevo paradigma de generación de aplicaciones. |

### API de servicios web.

Ésta nos va a permitir el intercambio de información entre diferentes servicios web y una aplicación, usando para ello el protocolo http o https y empleando los dos formatos más extendidos: XML o JSON.

Normalmente entre los desarrolladores vamos a encontrar cuatro tipos de API para servicios web:

* **XML-RPC** (*Extensible Markup Language – Remote Procedure Call*): protocolo que permite la invocación de procedimientos o funciones alojadas en un sistema remoto. Con este mecanismo los desarrolladores pueden definir interfaces que pueden ser invocadas a través de la red.
  + Emplea protocolo http para la transmisión de información desde el cliente hasta el equipo servidor.
  + Especifica las llamadas a las funciones o procedimientos a través de XML.
  + Emplea un “vocabulario XML” muy limitado para realizar las llamadas (*requests*) y recibir las respuestas (*responses*).
  + El cliente XML-RPC especifica unos parámetros en la llamada y recibe del servidor una respuesta en formato XML o un error, en caso de producirse.
  + Las primeras definiciones del protocolo proceden del año 1998.
* **SOAP** (*Simple Object Access Protocol*): protocolo estándar para el intercambio de mensajes en formato XML entre dos objetos y diseñado para las comunicaciones vía Internet.
  + Es una aplicación de la especificación XML.
  + Permite el transporte de datos por los servicios web.
  + No está acoplado a ninguna plataforma o lenguaje de desarrollo.
  + Permite a las aplicaciones cliente conectarse de manera sencilla a servicios remotos e invocar métodos remotos.
* **JSON-RPC** (*Javascript Object Notation- Remote Procedure Call)*: concepto similar al XML-RPC, para la llamada a procedimientos o funciones remotas, pero realizados en formato JSON.
  + Emplea el protocolo http para la transmisión de información.
  + Paso de nombre del procedimiento o función y de los parámetros necesarios desde el cliente en un objeto con formato JSON y respuesta desde el servidor en un objeto con el mismo formato.
* **RESTful** (*Representational State Transfer*): Son servicios web basados en arquitectura REST.
  + Los servicios web RESTful se utilizan para la creación de APIs de aplicaciones basadas en web.
  + Se implementan sobre protocolo http.
  + Un servicio web RESTful define un URI, Uniform Resource Identifier un servicio, proporciona representación de recursos como JSON y un conjunto de métodos http (GET, POST, PUT, DELETE).
  + Independiente de la plataforma, sistema operativo y lenguaje de programación.

Con estos cuatro modelos de APIs podremos desarrollar diferentes capas de abstracción que permitan la interconexión de cada uno de los sistemas pertenecientes a cada comunidad autónoma.

### Blockchain

# Estado del arte.



## 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. Información accesible pero inmutable.



### 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 tiene 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***. En capítulos posteriores, y después de haber introducido algunos conceptos de criptografía, profundizaremos en este concepto.
* **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.

### ¿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 incluye (cuyo número viene determinado por diferentes factores).*
* *Otro código alfanumérico que enlazará con el siguiente bloque.”*

La imagen siguiente (*Ilustración 2*) muestra el concepto de una “cadena de bloques”.

Ilustración 2: Representación de la cadena de bloques.

**Block Hash: 0000078ecba1500**

**Previous Block Hash: 0000096effa4600**

**Block Hash: 0000011aee0534e**

**Previous Block Hash: 0000078ecba1500**

**Block Hash: 000056abff1189a**

**Previous Block Hash: 0000011aee0534e**

**Transaction Hash: 000aa76effa4133**

**Transaction Hash: aaf4000faee3a00**

**Bloque 76**

**Bloque 77**

**Bloque 78**

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. De modo que tendríamos la siguiente estructura (*Ilustración 3*). Este bloque se forma en el momento de la creación de la Blockchain.

Y está definido en un fichero en formato JSON que recibe el nombre de *genesis.json* (*Ilustración 4*).

Ilustración 3: Ejemplo de bloque genesis.json

|  |
| --- |
| { |

|  |
| --- |
| "config": { |

|  |
| --- |
| "chainId": 0, |

|  |
| --- |
| "homesteadBlock": 0, |

|  |
| --- |
| "eip155Block": 0, |

|  |
| --- |
| "eip158Block": 0 |

|  |
| --- |
| }, |

|  |
| --- |
| "alloc"   : {}, |

|  |
| --- |
| "coinbase"   : "0x0000000000000000000000000000000000000000", |

|  |
| --- |
| "difficulty" : "0x20000", |

|  |
| --- |
| "extraData"  : "", |

|  |
| --- |
| "gasLimit"   : "0x2fefd8", |

|  |
| --- |
| "nonce"   : "0x0000000000000042", |

|  |
| --- |
| "mixhash" : "0x0000000000000000000000000000000000000000000000000000000000000000", |

|  |
| --- |
| "parentHash" : "0x0000000000000000000000000000000000000000000000000000000000000000", |

|  |
| --- |
| "timestamp"  : "0x00" |

|  |
| --- |
| } |

### ¿Dónde están los bloques?

Los bloques se sitúan en los NODOS. Estos no son más que los ordenadores que componen la red, en donde reside un software, que se instala

**Block Hash: 00000aa11ba1500**

**Previous Block Hash: 0x0000**

**Block Hash: 0000011aee0534e**

**Previous Block Hash: 00000aa11ba1500**

**Transaction Hash: 000aa76effa4133**

**Transaction Hash: aaf4000faee3a00**

**Bloque 0**

**Bloque 1**

…

Ilustración 4: Bloque Genesis

### Criptografía de clave pública

## IPFS

IPFS es el acrónimo de Inter Planetary File System, y según consta en su web[[3]](#footnote-3)

# Petchain. Especificación de requisitos.



Petchain es un proyecto destinado al sector veterinario que permite la gestión de los sistemas de identificación de mascotas. 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.

## 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.

## 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.

## Participantes en el proyecto



### 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 1: 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 2: 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 3: 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 4: Dirección facultativa y Veterinaria

## 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](http://www.unir.net/) |

Tabla 5: 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 6: Organización 2

## Descripción del sistema



### Perspectiva del producto

Petchain es una aplicación web, que 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 posiblidad 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 en una nueva implementación de modelo de base de datos universal, el IPFS.

### 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.

### 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.

### Restricciones

Petchain es un proyecto web que debe ser posible su visualización 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.

## Descripción técnica del sistema



### 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).

### 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 entono 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.

Aunque no se exija el almacenamiento de credenciales en una base de datos, si es necesario que la aplicación tenga implementados sistemas de protección de URL mediante JWT, así como la validación de usuarios mediante nombre de usuario y contraseña.

### 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.

### 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 NodeJS.

## Objetivos del sistema

|  |  |
| --- | --- |
| **OBJ-0001** | **Alta de mascotas** |
| **Versión** | 1.0 (04/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa * Pedro Escario Bajo |
| **Descripción** | El sistema deberá permitir el alta de una mascota en el sistema, asociándola a un código de chip. Este es un identificador con logitud fija de quince (15) caracteres alfanumérico. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Vital |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 7: Objetivo de Alta de mascotas

|  |  |
| --- | --- |
| **OBJ-0002** | **Alta de dueños de mascotas** |
| **Versión** | 1.0 (04/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa * Pedro Escario Bajo |
| **Descripción** | El sistema deberá permitir el dar de alta a una persona, asociándola a una mascota. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Vital |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 8: Objetivo Alta de dueños.

|  |  |
| --- | --- |
| **OBJ-0005** | **Modificación de datos** |
| **Versión** | 1.0 (04/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa * Pedro Escario Bajo |
| **Descripción** | El sistema deberá permitir modificar los datos asociados a las mascotas y a sus dueños. el veterinario modificador deberá pagar un canon fijo en weis al veterinario identificador. |
| **Sub-objetivos** | * [OBJ-0006] Modificación de datos de mascotas: El sistema deberá permitir la modificación de los datos sanitarios de las mascotas, así como la inclusión de las vacunaciones periódicas obligatorias. * [OBJ-0007] Modificación de datos de los dueños: El sistema deberá permitir cambiar los datos correspondientes al dueño de la mascota, incluido el cambio a un nuevo propietario. |
| **Importancia** | importante |
| **Urgencia** | inmediatamente |
| **Estado** | Por definir |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 9: Objetivo de Modificación de datos

|  |  |
| --- | --- |
| **OBJ-0008** | **Baja de mascotas** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa |
| **Descripción** | El sistema deberá permitir dar de baja a una mascota del sistema, por motivos de defunción. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Por definir |
| **Estado** | Por definir |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 10: Obetivo de Baja de mascotas

|  |  |
| --- | --- |
| **OBJ-0009** | **Consulta de saldo** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá permitir consultar el saldo en ethers o weis que cada veterinario participante en el sistema, tiene por realizar acciones dentro del mismo. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Por definir |
| **Estado** | Por definir |
| **Estabilidad** | **Por definir** |
| **Comentarios** | Ninguno |

Tabla 11: Objetivo de Consulta de saldo de ETH

|  |  |
| --- | --- |
| **OBJ-00011** | **Control de acceso** |
| **Versión** | 1.0 (05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá permitir el acceso al sistema a aquellos usuarios que estén incluidos en el fichero de usuarios creado |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | Pendiente de validación |
| **Estabilidad** | Alta |
| **Comentarios** | Ninguno |

Tabla 12: Objetivo de control de acceso al sistema

|  |  |
| --- | --- |
| **OBJ-00012** | **Visualización de múltiples dispositivos** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá estar diseñado para poder visualizarse en dispositivos del tipo Smartphone, Tablet y PC |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Por definir |
| **Estado** | Por definir |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 13: Objetivo de visulaización en múltiples disposivos.

|  |  |
| --- | --- |
| **OBJ-00013** | **Traspasar ETH** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá traspasar ETH desde el veterinario modificador hasta el veterinario identificador, siempre y cuando el modificador realice un cambio en los datos de una mascota que ya exista en el sistema. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 14: Objetivo de traspaso de ETH

|  |  |
| --- | --- |
| **OBJ-00014** | **Navegación por la aplicación** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá permitir la navegación entre las opciones de alta, modificación y baja de mascotas, así como la consulta del saldo de weis. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Vital |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | **Por definir** |
| **Comentarios** | Ninguno |

Tabla 15: Objetivo de la navegación de la aplicación

|  |  |
| --- | --- |
| **OBJ-00015** | **Gestión de sesiones** |
| **Versión** | 1.0 (05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Descripción** | El sistema deberá ser capaz de manejar las sesiones de usuario y el correcto funcionamiento de los navegadores. |
| **Sub-objetivos** | Ninguno |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | **Por definir** |
| **Comentarios** | Ninguno |

Tabla 16: Objetivo de Gestión de sesiones

## Catálogo de requisitos del sistema.



### 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** | * Carlos Escario Bajo | |
| **Fuentes** | * ? | |
| **Dependencias** | * [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** | * Nombre de usuario * Contraseña del usuario * Rol * Número de colegiado * Colegio | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Ocurrencias simultáneas** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Importancia** | Vital | |
| **Urgencia** | Por definir | |
| **Estado** | Por definir | |
| **Estabilidad** | Por definir | |
| **Comentarios** | Ninguno | |

Tabla 17: Requisito de Informacion del veterinario

|  |  |  |
| --- | --- | --- |
| **IRQ-0002** | **Datos de mascotas** | |
| **Versión** | 1.0 (05/05/2019) | |
| **Autores** | * Carlos Escario Bajo | |
| **Fuentes** | * 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** | * Nº 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** |
| Por definir | Por definir |
| **Ocurrencias simultáneas** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Importancia** | Por definir | |
| **Urgencia** | Por definir | |
| **Estado** | Por definir | |
| **Estabilidad** | Por definir | |
| **Comentarios** | Ninguno | |

Tabla 18: Requisitos de información de mascotas

|  |  |  |
| --- | --- | --- |
| **IRQ-0003** | **Datos del dueño** | |
| **Versión** | 1.0 (05/05/2019) | |
| **Autores** | * Carlos Escario Bajo | |
| **Fuentes** | * 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** | * Nombre * Apellidos * Número de documento * Teléfono * Provincia * Municipio * Código Postal * País | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Ocurrencias simultáneas** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Importancia** | Por definir | |
| **Urgencia** | Por definir | |
| **Estado** | Por definir | |
| **Estabilidad** | Por definir | |
| **Comentarios** | Ninguno | |

Tabla 19: Requisitos de información de propietarios.

|  |  |  |
| --- | --- | --- |
| **IRQ-0004** | **Saldo** | |
| **Versión** | 1.0 (05/05/2019) | |
| **Autores** | * Carlos Escario Bajo | |
| **Fuentes** | * 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** | * Cuenta Ethereum * Saldo en weis * Número de colegiado | |
| **Tiempo de vida** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Ocurrencias simultáneas** | **Medio** | **Máximo** |
| Por definir | Por definir |
| **Importancia** | Importante | |
| **Urgencia** | Por definir | |
| **Estado** | Por definir | |
| **Estabilidad** | Por definir | |
| **Comentarios** | Ninguno | |

Tabla 20: Requisitos de información del saldo en ETH

### 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, pags 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** | * Carlos Escario Bajo |
| **Fuentes** | * 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** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 21: Requisito funcional de acceso al sistema

|  |  |
| --- | --- |
| **FRQ-0002** | **Identificación de veterinarios** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá mostrar el nombre del veterinario y el colegio en donde está inscrito, con solo introducir el número de Colegiado |
| **Importancia** | Importante |
| **Urgencia** | Puede esperar |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 22: rquisito funcional de identificación de verinarios.

|  |  |
| --- | --- |
| **FRQ-0003** | **Traspaso de weis** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá traspasar weis 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** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 23: Requisito funcional de traspaso de weis

|  |  |
| --- | --- |
| **FRQ-0004** | **Navegación** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Pedro Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá permitir en todo momento la navegación por el sistema a través de las opciones que se muestren en la pantalla de inicio o en un menú situado en la cabecera de la aplicación. |
| **Importancia** | Vital |
| **Urgencia** | Imediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 24: Requisito funcional de navegación en el sistema.

|  |  |
| --- | --- |
| **FRQ-0005** | **Modificación de datos de una mascota** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa |
| **Dependencias** | Ninguno |
| **Descripción** | Como veterinario quiero poder modificar los datos sanitarios de una mascota, en concreto:   * Peligrosidad. * Nº de pasaporte. * Aptitud. * Fecha de revisión. * Estado. * Cicatrices. * Vacunas. * Observaciones |
| **Importancia** | Vital |
| **Urgencia** | Puede esperar |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 25: Requisito funcional de modificación de datos de una mascota

|  |  |
| --- | --- |
| **FRQ-0006** | Baja de una mascota |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá Como veterinario quiero poder marcar una mascota como fallecida. En concreto indicar:   * Causa del fallecimiento. * Número del veterinario. * Nombre del veterinario. * Apellidos del veterinario. |
| **Importancia** | Importante |
| **Urgencia** | Puede esperar |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 26: Requisito funcional para la baja de una mascota

|  |  |
| --- | --- |
| **FRQ-0007** | **Consultar weis disponibles** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | 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. , |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 27: Requisito funcional para la visualización de weis disponibles

|  |  |
| --- | --- |
| **FRQ-0009** | **Alta de una mascota** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | 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:   * 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** | Por definir |
| **Urgencia** | Por definir |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 28: Requisito funcional para el alta de una mascota.

|  |  |
| --- | --- |
| **FRQ-0010** | **Alta de propietario** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa * Pedro Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá permitir al veterinario dar de alta al dueño de una mascota, en la página de "Alta de Mascotas", que contenga los campos imprescindibles para identificar al dueño de un animal, en concreto sería necesarios:   * Colegio del veterinario. |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 29: Requisito funcional para el alta de un propietario.

|  |  |
| --- | --- |
| **FRQ-0011** | Modificación de los datos de un propietario |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Iciar García-Ochoa * Pedro Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | Como veterinario quisiera poder modificar los datos del dueño de una mascota, en la página de "Modificación de Mascotas", que contenga los campos imprescindibles para identificar al dueño de un animal, en concreto sería necesarios:   * Nombre. * Apellidos. * Dirección: Calle, Ciudad, CP y País. * Id del documento de identificación. * Teléfono. |
| **Importancia** | Importante |
| **Urgencia** | Inmediatamente |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 30: Requisito funcional para la modificación de un propietario.

### 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* (pag. 87).

|  |  |
| --- | --- |
| **NFR-0001** | **Visualización** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * 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** | Puede esperar |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 31: Requisito no funcional de visualización de Petchain.

|  |  |
| --- | --- |
| **NFR-0002** | **Moneda de pago** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá permitir el pago en weis, siendo la conversión en ethers de 1 ether = 1 e-18 wei. |
| **Importancia** | Importante |
| **Urgencia** | Por definir |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 32: : Requisito no funcional del pago en weis

|  |  |
| --- | --- |
| **NFR-0003** | **Tiempo de sesión** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * 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** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 33: Requisito no funcional del tiempo de sesión.

|  |  |
| --- | --- |
| **NFR-0004** | **Usuarios autorizados** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * 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 34: : Requisito no funcional de usuarios autorizados

|  |  |
| --- | --- |
| **NFR-0005** | **Número de cuentas** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * 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** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 35: Requisito no funcional del nº de cuentas en Ganache

|  |  |
| --- | --- |
| **NFR-0006** | **Autenticación JWT** |
| **Versión** | 1.0 ( 05/05/2019 ) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * Carlos Escario Bajo |
| **Dependencias** | Ninguno |
| **Descripción** | El sistema deberá tener un control de autenticación mediante JWT, que se almacena como variable en la SessionStorage. |
| **Importancia** | Importante |
| **Urgencia** | Puede esperar |
| **Estado** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 36: Requisito no funcional para la autenticación JWT

|  |  |
| --- | --- |
| **NFR-0007** | **Campos obligatorios** |
| **Versión** | 1.0 (05/05/2019) |
| **Autores** | * Carlos Escario Bajo |
| **Fuentes** | * 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** | En construcción |
| **Estabilidad** | Por definir |
| **Comentarios** | Ninguno |

Tabla 37: Requisito no funcional de campos obligatorios

## Matriz de rastreabilidad



### Objetivos del sist. – Requisitos funcionales / Requisitos no funcionales

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TRM0001** | **FRQ0001** | **FRQ0002** | **FRQ0003** | **FRQ0004** | **FRQ0005** | **FRQ0006** | **FRQ0007** | **FRQ0009** | **FRQ0010** | **FRQ0011** | **NFR0001** | **NFR0002** | **NFR0003** | **NFR0004** | **NFR0005** | **NFR0006** |
| **OBJ0001** | - | - | - | - | - | - | - |  | - | - | - | - | - | - | - | - |
| **OBJ0002** | - | - | - | - | - | - | - | - |  | - | - | - | - | - | - | - |
| **OBJ0005** | - | - |  | - | - | - |  | - | - | - | - | - | - | - | - | - |
| **OBJ0006** | - | - | - | - |  | - | - | - | - | - | - | - | - | - | - | - |
| **OBJ0007** | - | - | - | - | - | - | - | - | - |  | - | - | - | - | - | - |
| **OBJ0008** | - | - | - | - | - |  | - | - | - | - | - | - | - | - | - | - |
| **OBJ0009** | - | - | - | - | - | - |  | - | - | - | - | - | - | - | - | - |
| **OBJ0011** |  | - | - | - | - | - | - | - | - | - | - | - | - |  | - |  |
| **OBJ0012** | - | - | - | - | - | - | - | - | - | - |  | - | - | - | - | - |
| **OBJ0013** | - | - |  | - | - | - | - | - | - | - | - |  | - | - | - | - |
| **OBJ0014** | - | - | - |  | - | - | - | - | - | - | - | - | - | - | - | - |
| **OBJ0015** | - | - | - | - | - | - | - | - | - | - | - | - |  | - | - | - |

Tabla 38. Matriz de dependencias de requisitos

## 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.



### Gestión de mascotas

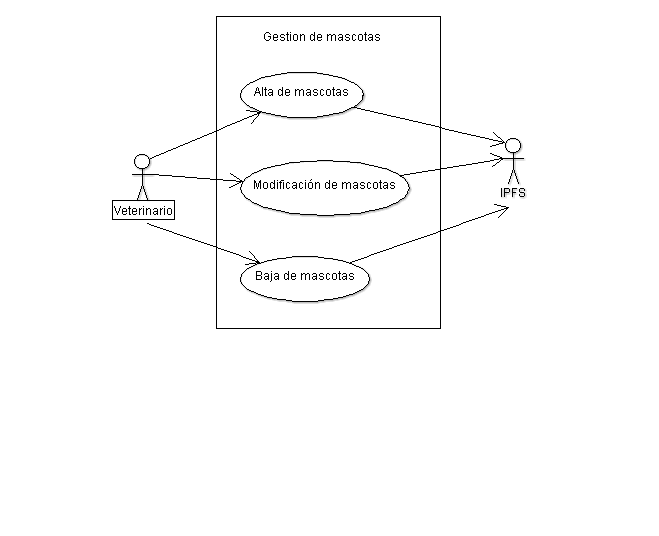


Ilustración 5: Gestión de mascotas

En la Ilustración 5 se expone como el veterinario podrá realizar las acciones de alta, modificación y baja sobre los datos de una mascota, y su posterior almacenamiento en la red IPFS. Tanto el alta de mascotas como la modificación, son procesos que a nivel de gráfico están integrados dentro del mismo proceso que la Gestión de propietarios, pero en la presente implementación se consideran Casos de Uso separados, pues su gestión de almacenamiento se llevará a cabo de manera separada dentro de la IPFS.

### Gestión de propietarios

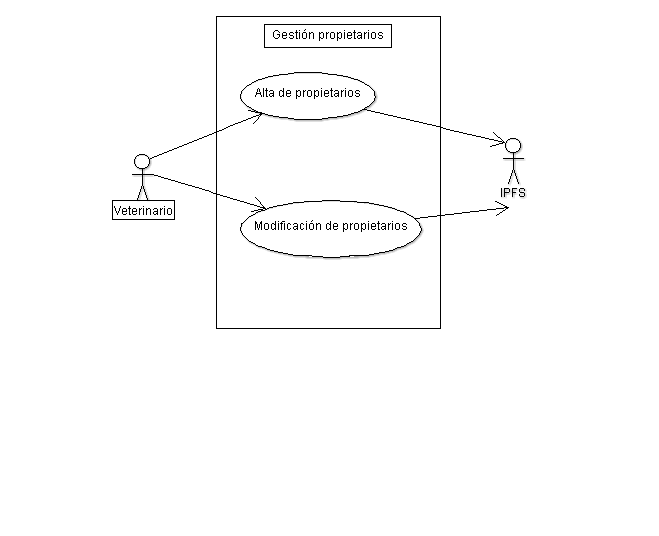


Ilustración 6: Gestión de propietarios

En este Caso de Uso se expone la gestión de los propietarios de las mascotas, y que como se observa sólo es posible realizar el alta y la modificación. Esto se debe a que un propietario siempre estará relacionado con una mascota, y las acciones siempre se realizarán sobre ésta, no sobre él, por lo que el sistema nunca permitirá el darlo de baja.

### Consulta de saldo

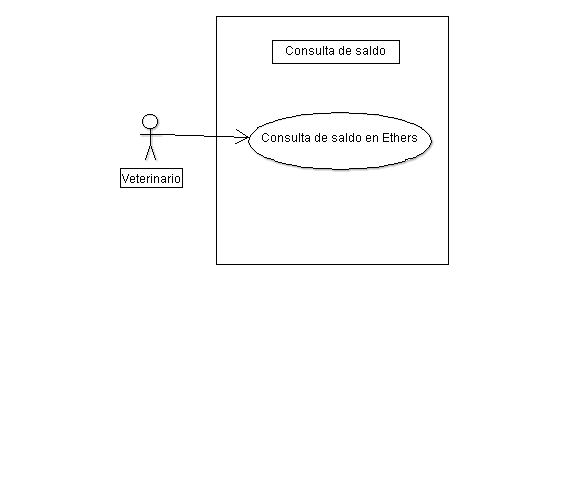


Ilustración 7: Consulta del saldo en Ethers

Este caso de uso muestra la consulta del saldo en moneda Ethers. Cada veterinario recibirá la cantidad de 0,1 ETH de otro veterianario 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 sistema.

# Metodología y Plan de trabajo

La metodología utilizada para el desarrollo de proyecto es el modelo evolutivo - iterativo..POR REDACTAR

## Seguimiento del proyecto

El seguimiento del proyecto ha sido realizado con la herramienta Trello[[4]](#footnote-4).

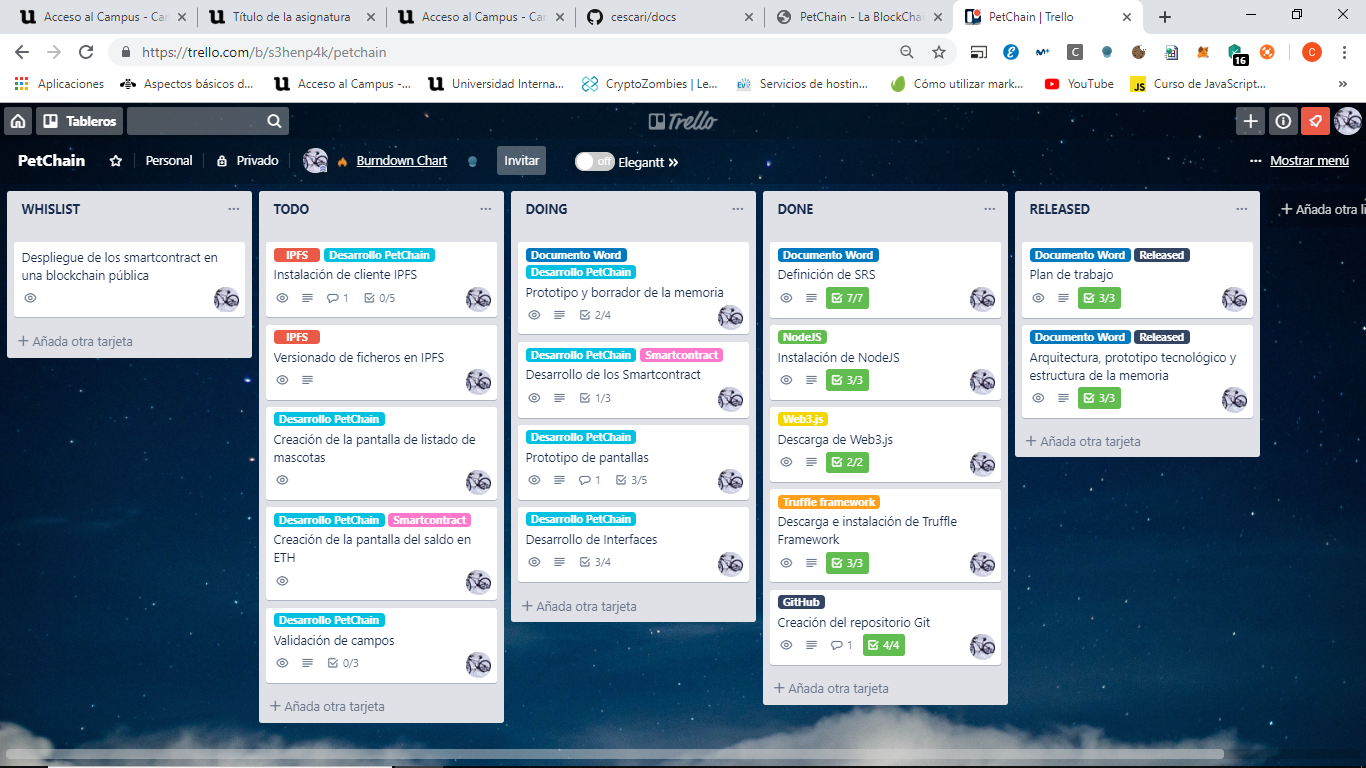


Ilustración 8: Tablero principal de Petchain en Trello

# Petchain. Descripción técnica.

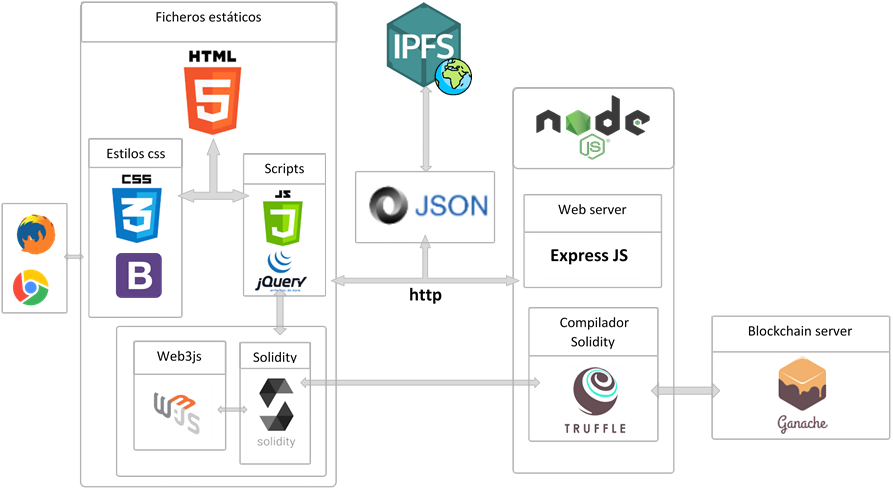
Petchain es una aplicación web que emplea para su ejecución una combinación de tecnologías, con el objetivo de conectar un frontal en HTML con un backend desarrollado con Smartcontracts e IPFS.

Ilustración 9: Esquema gráfico de Petchain (Fuente: diseño propio)

En los próximos apartados se van a introducir y explicar con detalle, por orden de instalación, cada una de las tecnologías necesarias para el correcto funcionamiento de la aplicación, así como una descripción funcional de cada uno de los elementos.



## NodeJS

Node.js es un entorno multiplataforma de código abierto 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 NodeJS 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. Para la instalación de dichos módulos se emplea un gestor de paquetes incluido en NodeJS llamado npm (Node Package Manager)[[5]](#footnote-5). Petchain para su ejecución necesita los paquetes descritos en las secciones siguientes.

|  |
| --- |
| **ATENCION**: Todas las instalaciones hechas vía **npm**, son realizadas con el parámetro **--save-dev**. Este parámetro indica que todos los módulos se deben instalar como una dependencia dentro del fichero **package.json**, en la lista de dependencias. Posteriormente, si se ejecuta el comando **npm install** en el directorio de la aplicación, los módulos se instalarán de manera automática en la lista de dependencias. |



### ExpressJS

Framework que permite la creación y ejecución de un servidor web dentro del entorno NodeJS.

|  |
| --- |
| **> npm install express -–save-dev** |

Ilustración 10: Comando instalación de ExpressJS

### body-parser

Módulo que permite la lectura de la cadena que se transmite en cualquier petición http entrante al servidor.

|  |
| --- |
| **> npm install body-parser –save-dev** |

Ilustración 11: Comando de instalación de body-parser

### jsonwebtoken

Estándar para la transmisión de información segura como un objeto JSON.

|  |
| --- |
| **> npm install express-jwt –save-dev** |

Ilustración 12: Comando de instalación de jsonwebtoken

### Generación del proyecto

La creación del proyecto NodeJS es un proceso sencillo, que no difiere del sistema operativo en el que se ejecute, y se realiza en los siguientes pasos.

|  |
| --- |
| **> mkdir tfg\_project**  **> cd tfg\_project** |

Ilustración 13: Comandos de creación directorio del proyecto

Creamos el fichero package.json, a través del comando npm init:

|  |
| --- |
| **> npm init** |

Ilustración 14: Inicialización del proyecto NodeJS

Al finalizar el proceso, se habrá creado un fichero package.json, que actuará como puerta de entrada para el proyecto. En nuestro caso, con las dependencias ya instaladas, éste archivo tendrá la siguiente estructura:

|  |
| --- |
| **{**  **"name": "tfg\_project",**  **"version": "1.0.0",**  **"description": "BlockChain para la gestión de un servicio de identificación de mascotas domésticas a nivel nacional.",**  **"main": "index.js",**  **"scripts": {**  **"test": "echo \"Error: no test specified\" && exit 1"**  **},**  **"repository": {**  **"type": "git",**  **"url": "git+https://github.com/cescari/tfg\_project.git"**  **},**  **"keywords": [**  **"tfg",**  **"blockchain",**  **"mascotas"**  **],**  **"author": "Carlos Escario Bajo",**  **"license": "SEE LICENSE IN ./license.html",**  **"bugs": {**  **"url": "https://github.com/cescari/tfg\_project/issues"**  **},**  **"homepage": "https://github.com/cescari/tfg\_project#readme",**  **"devDependencies": {**  **"body-parser": "^1.19.0",**  **"express": "^4.16.4",**  **"jsonwebtoken": "^8.5.1",**  **"nodemon": "^1.19.0",**  **"web3": "^1.0.0-beta.54"**  **},**  **"dependencies": {**  **"openurl": "^1.1.1"**  **}**  **}** |

Ilustración 15: Estructura del fichero package.json

## Truffle

Truffle[[6]](#footnote-6) 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 NodeJS, 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 NodeJS, por lo tanto para su instalación haremos uso de npm (Node Package Manager). Ejecutamos entonces la sentencia siguiente:

|  |
| --- |
| **> npm install –g truffle –save-dev** |

Ilustración 16: : 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.

**-g**: permite la instalación a nivel global del sistema.

### 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 17: 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 (\*.sol), su compilación y despliegue de binarios. Dicha estructura consta de los siguientes elementos.

* contracts/: directorio para el almacenamiento de ficheros solidity (\*.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.

|  |
| --- |
| **ATENCION**: 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. |

## Ganache

Ganache[[7]](#footnote-7) es un componente multiplataforma (Windows, Linux o Mac OS X), que forma parte del framework Truffle y su función es la servir como servidor de cuentas en una red Ethereum. Es el complemento ideal para el desarrollo de aplicaciones Blockchain en entornos cerrados, locales o corporativos, y está disponible para su ejecución como herramienta de escritorio o para invocarlo a través de línea de comandos (TestRPC).



### Instalación de Ganache

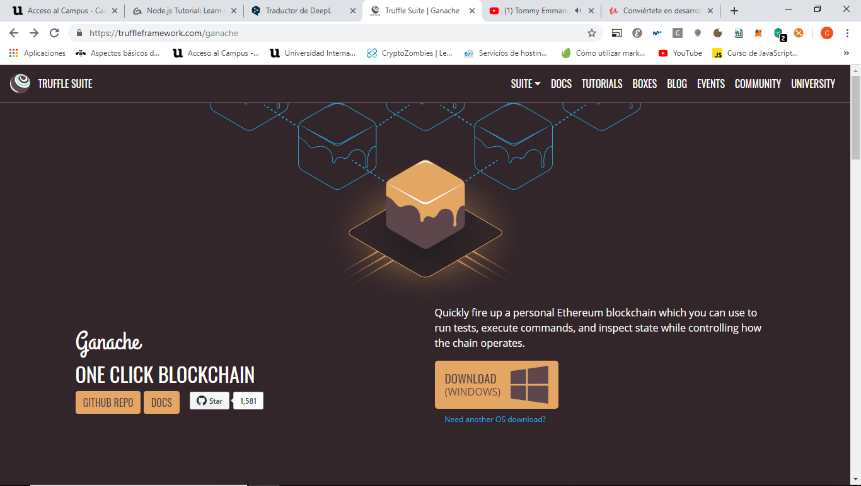
La instalación de Ganache es trivial y se realiza a través de un cásico “wizzard”. Para el presente proyecto se implementará como una herramienta de escritorio, tanto para un sistema Windows 10 y Linux Ubuntu 18.02, a través del instalador que se puede descargar desde la página web de Truffle.

Ilustración 18: Web de Truffle framework.

## 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 API[[8]](#footnote-8) 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 NodeJS a través del comando npm en la carpeta del proyecto, y se instalará como una dependencia del proyecto dentro del package.json de NodeJS.
* También es posible tener disponible la librería cuando se instala el framework de Truffle, dado que está incorporado como parte del paquete. El problema es que la versión disponible en la actualidad de Web3.js al instalar Truffle es la versión 0.23.0, la cual tiene bastantes problemas en cuanto a funcionalidades deprecadas.
* Como librería estática (692k) dentro del proyecto como: /src/js/vendor/web3.js, descargándolo de: [*https://github.com/ethereum/web3.js/releases/tag/v1.0.0-beta.54*](https://github.com/ethereum/web3.js/releases/tag/v1.0.0-beta.54).

Esta última opción, ha sido el método seleccionado para su incorporación dentro de Petchain.

## JQuery

JQuery, es quizás, una de las herramientas de desarrollo de scripts más conocidas del mercado, tanto por su extensión como por tiempo de funcionamiento. No se considera necesario, en el presente trabajo, realizar ningún tipo de introducción técnica dado su popularidad.

La herramienta se ha descargado como un fichero estático, en su versión 3.3.1, minificada y posicionada dentro de la carpeta: *src/vendor/jquery/jquery.min.js*.

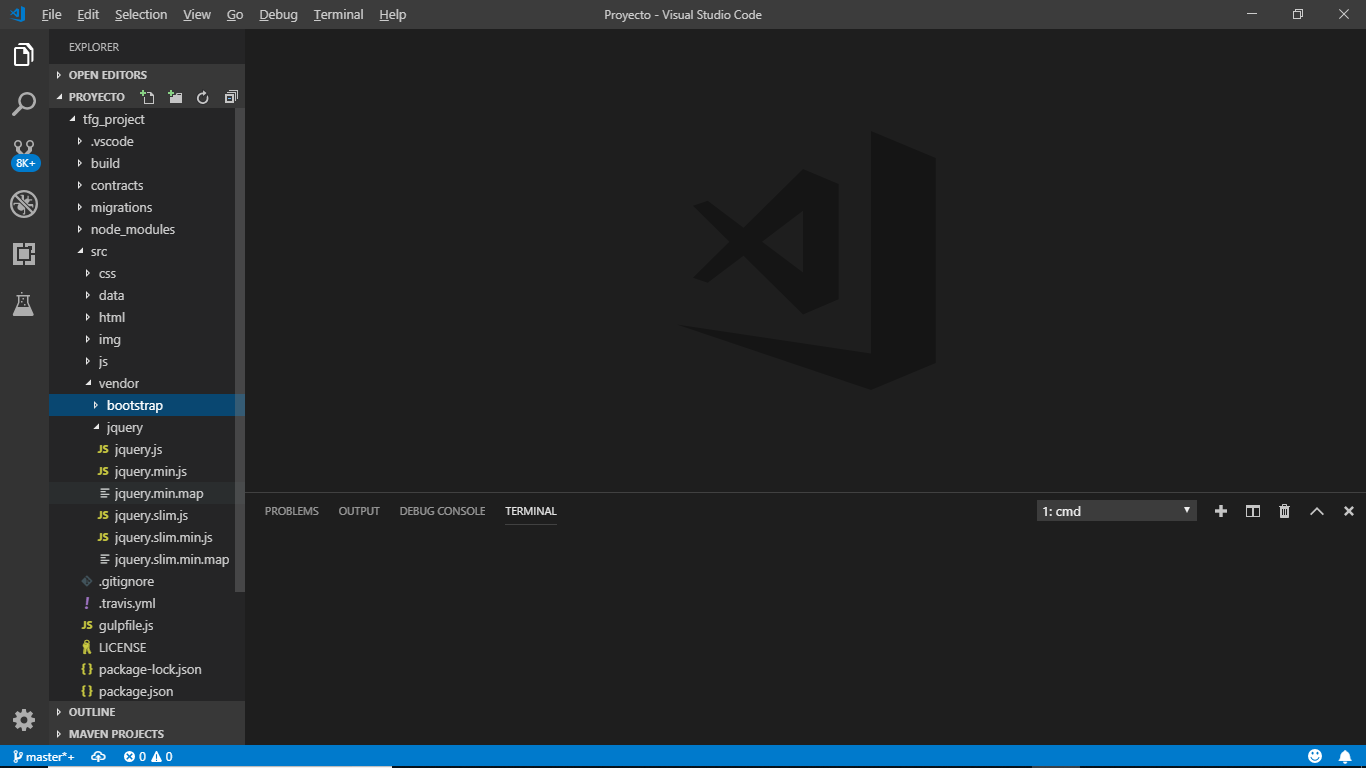


Ilustración 19: Estructura de vendor/jquery

## Bootstrap

Bootstrap[[9]](#footnote-9), 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.

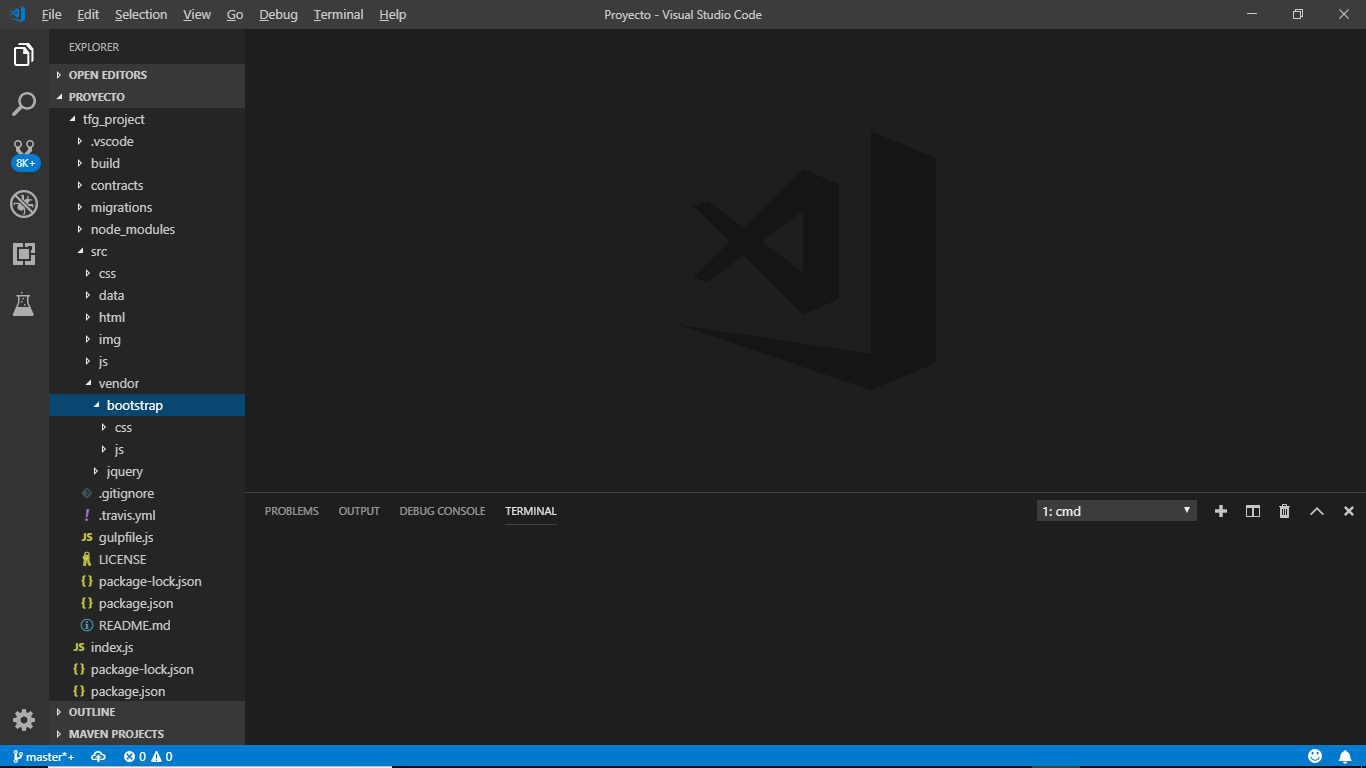


Ilustración 20: Estructura de vendor/bootstrap

## Herramientas utilizadas

## Patrón de diseño.

Petchain se ha diseñado bajo el patrón MVC, Modelo – Vista – Controlador. El desarrollo bajo este paradigma se basa en la separación del código en tres partes diferenciadas, con el objetivo de favorecer el mantenimiento del software generado.

Controlador

Vista

Modelo

Ilustración 21: Esquema clásico del patrón MVC. Fuente: Diseño propio

|  |  |
| --- | --- |
| **Relación Arquitectura- tecnologías utilizadas** | |
| **Vista** | HTML5 |
| css |
| JavaScript |
| **Controlador** | Javascript |
| **Modelo** | Solidity |
| JSON |
| Javascript |

Tabla 39: Relación Arquitectura - tecnología

### 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, como se indica en la Tabla 7, por el conjunto de ficheros *\*.sol* (solidity), los ficheros con estructuras JSON, y los archivos Javascript que componen los servicios, así como los necesarios para la compilación de los ficheros solidity.

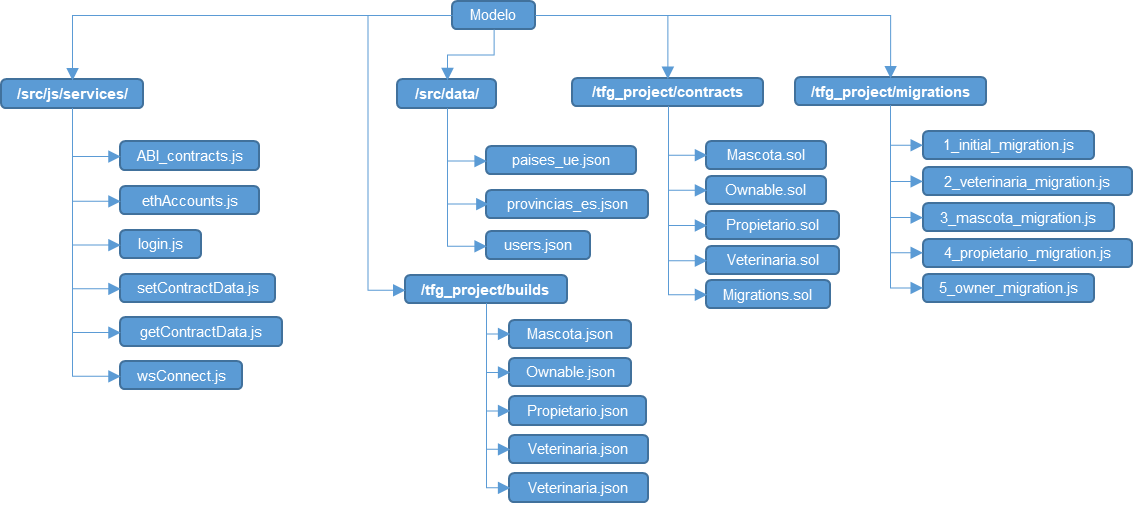


Ilustración 22: Estructura de los ficheros del Modelo

### Vista

La vista es el conjunto de ficheros que componen la parte gráfica y que determinan como debe de visualizarse la aplicación. Como aplicación “web clásica”, Petchain tiene los ficheros de estilos (\*.css) y html completamente separados, sin que exista en ninguno definiciones *in-line*, para evitar acoplamientos y facilitar el mantenimiento. El conjunto de ficheros que componen la estructura de la vista son los reflejados en la imagen siguiente (*Ilustración 22)*. En la estructura se han incluido los ficheros gráficos, que aunque no son elementos de la programación, si son parte fundamental de la vista.

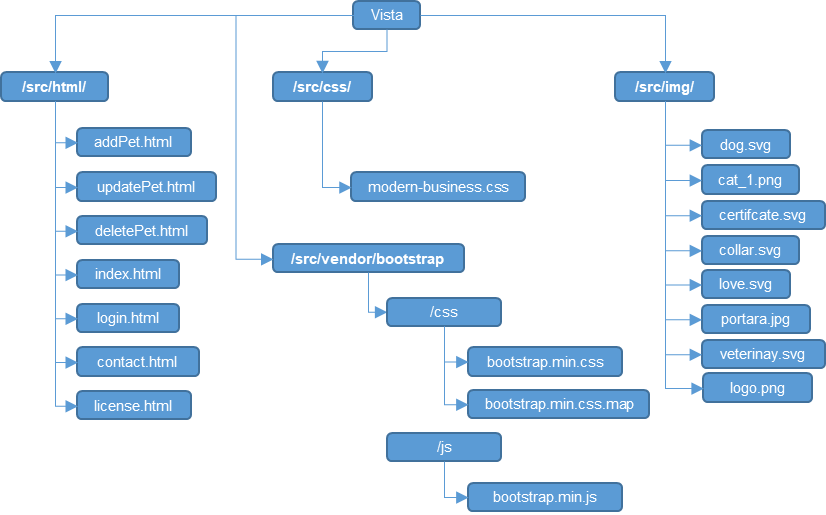


Ilustración 23: Estructura de ficheros de la Vista

### Controlador

Esta es la capa encargada de realizar las modificaciones realizadas por el modelo y/o la vista en respuesta a las peticiones realizas desde el frontal o la base de datos. Su objetivo es servir como capa de abstracción entre el modelo y la vista. En la aplicación se ha buscado optimizar la codificación, es decir hacerla lo más estándar posible entre ficheros, con el objeto de minimizar la codificación. Esto hasta el momento ha permitido que hasta ahora tan sólo sea necesaria la intervención de un solo fichero dentro la capa controladora.

Aun así su función es de suma importancia, pues en él 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.

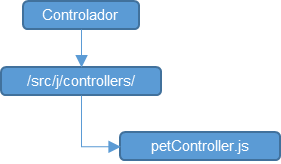


Tabla 40: Esquema del fichero que compone la capa controladora.

# Petchain. Descripción funcional

# Evaluación

# ¿Qué queda por hacer? Trabajo futuro.

# Bibliografía

|  |
| --- |
| 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 smart contracts 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. |

# Glosario de términos

*A*

*B*

*C*

CSS3

Cascade Style Sheet ver.3 17

*D*

DNI

Documento nacional de identidad 17

*E*

ECMA

European Computer Manufactures

Association 17

ETH

Ethers 6, 18, 19,

26, 30

*F*

*G*

*H*

HTML5

Hiper Text Markup Language 17

*I*

IPFS

Inter Planetary File System 1, 2,14,

15, 19, 34,35

*J*

JSON

Java Script Object Notation 4, 7,8,

11, 17, 30, 35, 49

JWT

Javascript Web Token 17,30

*K*

*L*

*M*

MVC

Model View Controller 17

*N*

NoSQL

No Structured Query Language 16,17

npm

Node package manager 41, 43, 45

*O*

*P*

P2P

Peer to peer 12

*Q*

*R*

*S*

*T*

*U*

UNIR

Universidad Internacional de la Rioja 1

URL

Unifrom Resource Locator 17

*V*

*X*

XML

Extensible Markup Language 10

*Y*

*Z*

# Referencias

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

1. [*http://petshopsmagazine.com/mascotas/censo-mascotas-espana/*](http://petshopsmagazine.com/mascotas/censo-mascotas-espana/) [↑](#footnote-ref-1)
2. *www.siacyl.org* [↑](#footnote-ref-2)
3. [*https://docs.ipfs.io/introduction/overview/*](https://docs.ipfs.io/introduction/overview/) [↑](#footnote-ref-3)
4. *Trello web site*: [*https://trello.com*](https://trello.com/) [↑](#footnote-ref-4)
5. *npm web site:* [*https://www.npmjs.com/*](https://www.npmjs.com/) [↑](#footnote-ref-5)
6. *Truffle website: https://truffleframework.com* [↑](#footnote-ref-6)
7. *Ganache website:* [*https://truffleframework.com/ganache*](https://truffleframework.com/ganache) [↑](#footnote-ref-7)
8. [*https://github.com/ethereum/wiki/wiki/JavaScript-API#web3-javascript-app-api-for-02xx*](https://github.com/ethereum/wiki/wiki/JavaScript-API#web3-javascript-app-api-for-02xx) [↑](#footnote-ref-8)
9. [*https://getbootstrap.com*/](https://getbootstrap.com/) [↑](#footnote-ref-9)