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.

PetChain es una aplicación “web clásica”, que permite la gestión de identidades de mascotas domésticas. Desarrollada como una Blockchain bajo una red [*Ethereum*](#_Referencias), funcionando en un entorno local en un servidor [*Ganaché*](#_Referencias) 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



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.

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 *Ganaché* 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, nodo.

Indice

[Resumen 1](#_Toc7626043)

[Summary 1](#_Toc7626044)

[Índice de ilustraciones y tablas 3](#_Toc7626045)

[1. Introducción 4](#_Toc7626046)

[1.1. Presentación del problema 4](#_Toc7626047)

[1.2. La solución. 5](#_Toc7626048)

[1.1.1. API de servicios web. 6](#_Toc7626049)

[1.1.2. Blockchain 7](#_Toc7626050)

[2. Estado del arte. La Blockchain 8](#_Toc7626051)

[2.1. Características 8](#_Toc7626053)

[2.2. ¿Qué es un bloque? 8](#_Toc7626054)

[2.3. ¿Dónde están los bloques? 10](#_Toc7626055)

[2.4. Criptografía de clave pública 10](#_Toc7626056)

[3. PetChain. Descripción del proyecto 10](#_Toc7626057)

[4. Identificación de requisitos. 10](#_Toc7626058)

[5. PetChain. Descripción técnica 11](#_Toc7626059)

[6. Evaluación 11](#_Toc7626060)

[7. ¿Qué queda por hacer? Trabajo futuro. 11](#_Toc7626061)

[8. Referencias 11](#_Toc7626062)

Índice de ilustraciones y tablas

[Ilustración 1: PetChain logo 1](#_Toc7560668)

[Ilustración 2: Representación de la cadena de bloques. 9](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc7560669)

[Ilustración 3: Bloque Genesis 9](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc7560670)

[Ilustración 4: Ejemplo de bloque genesis.json 10](file:///C:\Users\Carolina\Google%20Drive\UNIR\GRADO%20INGENIERIA%20INFORMATICA\TFG\docs\TFG_Carlos_Escario_Bajo.docx#_Toc7560671)

# 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.

Ilustración 3: Bloque Genesis

**Block Hash: 00000aa11ba1500**

**Previous Block Hash: 0x0000**

**Block Hash: 0000011aee0534e**

**Previous Block Hash: 00000aa11ba1500**

**Transaction Hash: 000aa76effa4133**

**Transaction Hash: aaf4000faee3a00**

**Bloque 0**

**Bloque 1**

…

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

Ilustración 4: 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

## Criptografía de clave pública

# PetChain. Descripción del proyecto

# Identificación de requisitos.

Después de varias entrevistas con los usuarios del actual sistema, se han identificado una serie de requisitos básicos que permitan la gestión de los datos.

## Requisitos funcionales

Acceso

# PetChain. Descripción técnica

# Evaluación

# ¿Qué queda por hacer? Trabajo futuro.

# Referencias

1. [*https://ethereum.org/*](https://ethereum.org/)
2. [*https://truffleframework.com/ganache*](https://truffleframework.com/ganache)

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)