



VERSION
FINAL

15-9-2020

Blockchain & BigData Canino



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE DE DOCUMENTOS

I.....	MEMORIA
II	ANEXO I: INTRODUCCIÓN AL MUNDO CANINO
III	ANEXO II: ARQUITECTURA DEL PROYECTO
IV	ANEXO III: DESARROLLO DE CHAINCODES
V.....	ANEXO IV: DESARROLLO DE API-REST
VI	ANEXO V: APLICACIÓN BIG DATA



Fac
Uni



Facultad de Estudios Estadísticos
Universidad Complutense de Madrid

ntic
master
revolucionamos la comunicación

VERSION
FINAL

15-9-2020

Blockchain & BigData Canino

MEMORIA



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE

Agradecimientos	2
Equipo y definición de tareas	2
Estado del arte: Introducción al mundo canino	3
CONCEPTOS BÁSICOS DEL MUNDO CANINO	3
• Identificación canina	3
• Libro genealógico	4
• Federaciones caninas	4
• Afijos de criadores	5
• Certificaciones	5
• Procesos de registro	5
FUNCIONALIDADES BÁSICAS A DESARROLLAR	6
• Blockchain canino	6
• Big Data Canino	6
CONCLUSIONES	6
Arquitectura del Proyecto: Descripción del Producto y Fundamentos Técnicos	7
ARQUITECTURA DEL PROYECTO	8
SCRIPTS DE CONFIGURACIÓN	9
ARCHIVOS DE CONFIGURACIÓN	9
PUESTA EN MARCHA DEL PROYECTO	10
• Red inicial sin TLS	10
• Red con TLS	10
• Red creada entre dos servidores	11
Chaincodes y funcionalidades del sistema	11
CHAINCODES SEGÚN ORGANIZACIÓN	11
FUNCIONES COMUNES DE LOS CONTRATOS	12
CHAINCODES	12
• PERSONAS	12
• AFIOS	12
• PERROS	12
• SOLICITUDES	13
• PERFILES	13
• VETERINARIOS	13
• MICROCHIPS	13
• VACUNAS	13
• RAZAS	14
• Otros posibles contratos inteligentes	14
API-REST	14
Desarrollo Big Data	15
OBJETIVOS PRINCIPALES	15
ANÁLISIS EXPLORATORIO DE LOS DATOS	16
CONFIGURACIÓN DEL SERVIDOR AMAZON LINUX	16
ESTUDIO DE LOS MEJORES MODELOS CNNs	16
DEFINICIÓN Y ENTRENAMIENTO DE LOS MODELOS	17
COMPARACIÓN DE LOS MODELOS Y SELECCIÓN DEL MODELO GANADOR	17
GENERACIÓN DE UN APACHE TOMCAT EN EL SERVIDOR DE AMAZON LINUX	18
PREDICCIONES CON LOS DATOS DE PRUEBA	19
CONCLUSIONES	19



Agradecimientos

A nuestras **familias**,
sin cuyo apoyo y paciencia este trabajo no habría llegado a su fin

A la Facultad de Estudios Estadísticos de la **UCM** de Madrid y
a **NTIC** Máster por haber hecho posible la realización de este Máster

A nuestro tutor, **Sergio Torres**, por haber aguantado
nuestras preguntas y haber resuelto nuestras dudas

Equipo y definición de tareas

Cristina Rodríguez Chamorro	Desarrollo aplicación Big Data
Daniel Lanzas Pellico	Desarrollo arquitectura Blockchain
Helena García Fernández	Desarrollo aplicación Big Data
José Bennani Pareja	Desarrollo Web y API-REST
Juan José Lucas de la Fuente	Desarrollo Web y API-REST
Unai Ares Icaran ...	Definición y desarrollo de algoritmos BD, Chaincodes y Estado del Arte

Estado del arte: Introducción al mundo canino

En la actualidad el mundo de las mascotas constituye un negocio multimillonario de aproximadamente 36.500 millones al año en toda Europa, donde hay 200 millones de mascotas en 80 millones de hogares. En España genera más de 1.000 millones anuales de negocio que generan más de 100.000 puestos de trabajo, entre indirectos y directos.

Según datos de la Red Española de Identificación de Animales de Compañía (REIAC) existen en el país más 13 millones de mascotas registradas, de las cuales más de la mitad corresponden a ejemplares caninos. La pureza de raza, la línea genealógica y los premios que han obtenido sus ancestros determinan mucho el precio, por lo que la verificación y certeza de estos datos son especialmente relevantes en este tipo de negocio donde existen muchos casos de personas que se aprovechan de la gente.

Actualmente existen multitud de organizaciones en diferentes países que gestiona las certificaciones del mundo canino, cada una encargándose de diferentes partes procesos (registros de afijos de criadores, camadas, propietarios, identificación canina, premios y títulos, ...) y en muchos casos de manera repetida, duplicando competencias, por no existir un registro unificado.

Con el fin de poder reunir a estos colectivos en un solo entorno compartido se plantea un desarrollo Blockchain que sirva para interactuar inicialmente entre:

- Criadores caninos
- Propietarios
- Veterinarios
- Federaciones caninas
- Jueces
- Sociedades y Clubes caninos



CONCEPTOS BÁSICOS DEL MUNDO CANINO

- Identificación canina

Este proceso lo realizan los veterinarios insertando un pequeño [microchip subcutáneo](#) del tamaño de un grano de arroz, en una capsula de cristal especial que contiene un transpondedor con un código único, siendo obligatorio en España con el objeto de evitar el abandono animal estando tipificado como delito en la [Ley orgánica 1/2015](#).





Existe un registro gestionado por cada [comunidad autónoma](#) así como una [Red Española de Identificación de Animales de Compañía \(REIAC\)](#), que en muchos casos no está actualizado.

- Libro genealógico

Registro, fichero o sistema informático donde se inscribe los perros de raza pura, mencionando sus ascendentes y descendientes, gestionado por organizaciones reconocidas oficialmente a través de eventos.



Para su inscripción es necesario demostrar que las tres últimas generaciones del ejemplar se encuentran inscritas en un libro de origen de especies o si este hecho no pudiera ser demostrado, a través un reconocimiento de raza por un juez especialista de la organización.

- Federaciones caninas

Estas organizaciones se aseguran de que los pedigüeys, los criadores y los jueces sean reconocidos mutuamente por todos los miembros que las integran. Organizan [concursos y exposiciones](#) de Morfología, Trabajo y disciplina, Rastreo y Agility, concediendo menciones y títulos a los mejores ejemplares, a través de jueces reconocidos por dichas organizaciones.

Existen numerosas federaciones a nivel nacional e internacional, por ejemplo:

- [Fédération Cynologique Internationale \(FCI\)](#)
- [Real Sociedad Canina Española \(RSCE\)](#)
- [The Kennel Club \(TKC\)](#)
- [Alianz canine Worldwide \(ACW\)](#)

Las sociedades y clubes de raza son organizaciones que colaboran con las federaciones para realizar diferentes gestiones a nivel local para el fomento de razas caninas o de alguna en especial. Preparan en coordinación con las federaciones concursos y exposiciones, por ejemplo:

- [Sociedad Canina de Bizkaia](#)
- [Sociedad Canina Montañesa](#)
- [Sociedad valenciana para el fomento de razas caninas](#)
- [Sociedad Canina Gallega](#)





- Afijos de criadores

Un afijo es una denominación o marca personal de un criador que lo registra a través de una sociedad o federación canina.

La concesión de un [afijo](#) autoriza a su titular o titulares a utilizarlo en la inscripción de camadas. Todos los titulares del afijo deben ser propietarios de la hembra, madre de la camada, para poder utilizar el afijo.

El afijo es propiedad exclusiva de la persona o colectividad que ha adquirido el derecho y es vitalicio. El propietario de un afijo puede aplicarlo a todos los ejemplares de cualquier raza de la que es criador. No puede aplicarse a un ejemplar un afijo diferente al de su criador. Nadie podrá ser titular de más de un afijo por raza.

Por ejemplo:

- Afijo: [Coramonte](#)
- Ejemplares: [Vanessa de Coramonte](#), [Gastón de Coramonte](#),

- Certificaciones

Es necesario estudiar el ciclo completo de registro de los perros desde el momento del nacimiento para detallar las certificaciones que debería proporcionar el proyecto blockchain a desarrollar, por ejemplo:

- Certificado de propietario de ejemplar y traspaso de propiedad
- Certificado de nacimiento de camadas
- Certificado de cesión temporal de hembra
- Certificado de afijo de criador
- Certificado de identificación canina (microchip)
- Certificado de la línea genealógica o pedigree del ejemplar
- Certificado de reconocimiento de raza del ejemplar
- Certificado premios y títulos del ejemplar

Procesos de registro

Los actuales procedimientos de las federaciones caninas se realizan mayoritariamente de manera presencial (ya que es necesario realizar pagos por las gestiones), lo que supone desplazarse hasta sus oficinas y pagar las correspondientes tasas.

La Blockchain permitirá certificar todos estos procesos puedan realizarse online, con una mayor seguridad desde cualquier punto del mundo, a cualquier hora.

FUNCIONALIDADES BÁSICAS A DESARROLLAR

- Blockchain canino

Las funcionalidades básicas a desarrollar en el proyecto Blockchain serán:

- Identificación y autenticación (Criadores caninos, propietarios, veterinarios, sociedades caninas, jueces, clubes caninos)
- Afijo de criador (alta, baja, modificación, certificación)
- Camadas (registro y certificación)
- Ejemplar (certificación propiedad y traspaso, pedigree y defunción)
- Microchip (registro y modificación)
- Registro de vacunación
- Concursos y exposiciones (registro de premios y títulos)

En una puesta en producción, podría realizarse las siguientes ampliaciones:

- Identificación a través de ADN
- Registro de enfermedades



- Big Data Canino

Existen algunas ideas sobre las ampliaciones que se podrían realizar a partir de la Base de Datos ya generada con Blockchain y nuevos datos acumulados:

- Calidad estimada del cruce de razas a partir de los modelos obtenidos por las líneas genealógicas y los premios y títulos conseguidos.
- Estudio de características morfológicas a partir del ADN y detección de enfermedades

Para esta prueba de concepto se ha optado por la implementación de :

- Reconocimiento de razas a través de imágenes (Deep learning)

CONCLUSIONES

El mundo canino esconde detrás una gran complejidad con múltiples participantes con intereses en algunas ocasiones en común y en otros particulares, sobre todo en cuanto comprendemos que detrás de las líneas genealógicas de los

perros, el prestigio, los concursos y exposiciones se mueve mucho dinero, lo que puede hacer que cada organización se muestre recelosa, para compartir su información, produciéndose en algunos casos perdida de información, datos duplicados o no incongruentes por falta de actualización.

Por otro lado, existen los amigos de lo ajeno que intentan engañar y aprovecharse de personas, falseando datos y cometiendo fraudes, en contraposición con algunas asociaciones cuyo fin es el fomento de las razas caninas, pero se ven limitados por no disponer de los recursos necesarios para abordar un proyecto global. Es por ello, que un blockchain canino podría resolver muchos de los problemas citados.

Arquitectura del Proyecto: Descripción del Producto y Fundamentos Técnicos

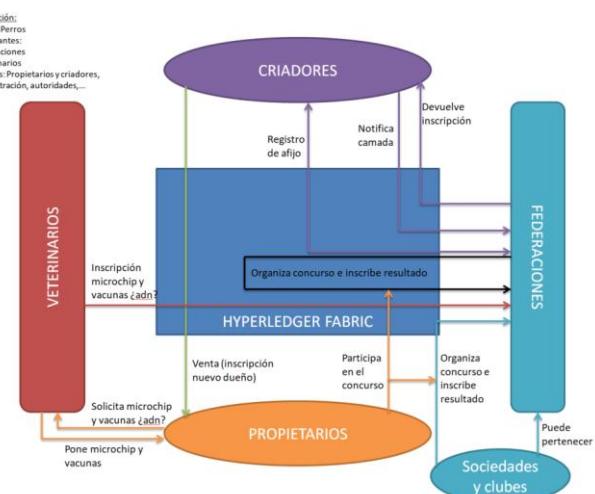
En primer lugar, se decide el uso de la tecnología Blockchain para regir las relaciones entre las diferentes organizaciones, lo que redunda en que la actividad sea transparente para ambas y en que la información sea inmutable y no pueda ser modificada para realizar ningún tipo de fraude.

Por otra parte, dado que se trata de dos organizaciones privadas que deben interactuar sin llegar a tener confianza una en la otra y que los datos tienen una componente privada (siendo algunos de ellos sensibles y protegidos por el RGPD), se propone la utilización de una red Blockchain privada con la tecnología Hyperledger Fabric como base.

Los activos con los que contará la red son los perros, que tendrán un propietario que podrá cambiar, se les podrá poner el microchip y vacunas, participarán en concursos,...

Los participantes serán las Federaciones Caninas y los Colegios Veterinarios, que son los que ejecutarán las acciones sobre los perros.

Por otra parte que los usuarios serán las propias Federaciones, los Colegios Veterinarios, los propietarios, los veterinarios y las autoridades competentes (agentes de la autoridad, juzgados,...) a los que se permita el uso de la red.



ARQUITECTURA DEL PROYECTO

Dentro de Hyperledger Fabric se definen dos organizaciones con las siguientes características:

- Federaciones Caninas

Se trata de las organizaciones que llevan a cabo el control de los afijos autorizados para la cría de perros, control de las características de las razas, pedigüeis,...

En este caso se tendrán en cuenta las cuatro mayores Federaciones existentes:

- FCI: Fédération Cynologique internationale
- RSCE: Real Sociedad Canina Española
- TKC: The Kennel Club
- ACW: Alianz Canine Worldwide

- Colegios Veterinarios

Se trata de las organizaciones que llevan el control de todo lo relativo a la sanidad de los perros: colocación de microchip, vacunas, pasaporte sanitario, ...

Se tomarán los Colegios Veterinarios a nivel de Comunidad Autónoma, por lo que tendremos diecisiete Colegios Veterinarios

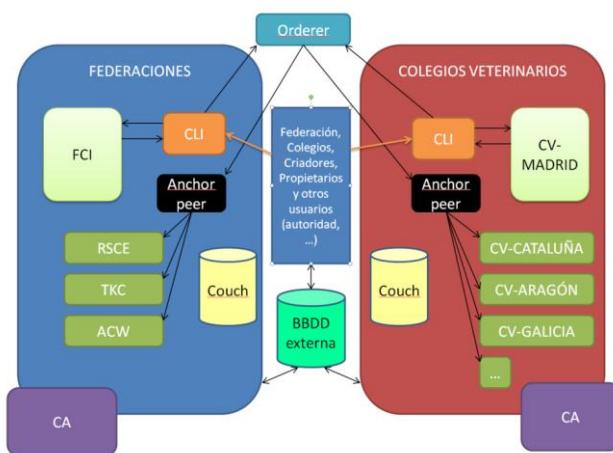
Se utilizará CouchDB en vez de levelDB para aumentar las posibilidades de consulta a la red y cada organización tendrá una autoridad de certificación (CA).

Además, se implementará una base de datos externa para guardar los datos sensibles y los destinados al uso por la aplicación de Big Data.

El tipo de consenso será “solo” para el desarrollo, aunque posteriormente debería implementarse “Kafka” en producción.

Una vez definidas las organizaciones la arquitectura será la siguiente:

Tras la instalación de Hyperledger Fabric en los servidores crearemos una red con la arquitectura descrita





anteriormente, consistente en dos organizaciones (Federaciones y Colegios Veterinarios), cuatro peers para la primera organización y diecisiete para la segunda, Orderer, CLI, CA's y CouchDB. Instalaremos además Hyperledger Explorer para poder visualizar la información de la red en el navegador, accediendo también a la CouchDB desde el navegador.

Posteriormente añadiremos TLS a la red, generando una red más segura en la que será necesario proveer los certificados correspondientes al efectuar cualquier acción.

Por último desplegaremos la red entre dos servidores simulando la existencia real de las dos organizaciones.

SCRIPTS DE CONFIGURACIÓN

Tras definirse la estructuras de carpetas del proyecto se decide realizar una serie de scripts que levanten y paren el proyecto de manera automática, tanto en su despliegue en un servidor como en dos servidores, siendo los siguientes scripts:

- netcan_script.sh: En primer lugar limpia la instalación, a continuación accede al repositorio de Github, descarga las carpetas json, chaincode y scripts y sustituye las existentes por estas y posteriormente va lanzando el resto de scripts de configuración, ya sea para uno o dos servidores
- red_script.sh: Este script levanta la red hyperledger Fabric y copia los scripts necesarios al docker del CLI, donde luego habrá que realizar la configuración del proyecto
- Serv2_script.sh: Este script realiza la configuración en el servidor 2 en el caso de que se levante la red entre dos servidores
- config_script.sh: Este script se ejecuta dentro del docker del CLI y realiza la configuración del canal, los peers y los pares de anclaje, ya sea con TLS o sin TLS
- chaincode_script.sh: Este script también se ejecuta dentro del docker del CLI e instala e instancia los chaincodes desarrollados
- json_script: Este script copia los archivos de carga iniciales a los dockers de los chaincodes para realizar las cargas iniciales de datos a la blockchain
- carga_script.sh: Este script se ejecuta dentro del docker del CLI y realiza las cargas iniciales de datos a la blockchain
- stop_netcan_script: Por último este script para la red, borra los dockers levantados y borra los chaincodes instanciados

ARCHIVOS DE CONFIGURACIÓN

- configtx.yaml: En este archivo se realiza la configuración de las organizaciones y del canal y es el que servirá para la generación de los artefactos del proyecto.

- **crypto-config.yaml:** Define la arquitectura de la red con el orderer, las organizaciones y los peers para la generación del material criptográfico por medio de la herramienta de hyperledger fabric cryptogen
- **docker-compose-cli.yaml:** Es el archivo de configuración principal de Hyperledger Fabric. En él se define la estructura básica del proyecto.
- **docker-compose-couch.yaml:** Este archivo configura la red para el uso de CouchDB en vez de levelDB, lo que permite realizar índices y redundar en una mayor facilidad de uso y la posibilidad de realización de consultas más complejas.
- **docker-compose-base.yaml:** En este archivo se definen el orderer y los peers. En particular se establece que los peers extienden a su vez del archivo peer-base.yaml y el uso de TLS en el orderer pasando los certificados correspondientes en las variables
- **peer-base.yaml:** Establece la configuración común a todos los peers, incluido el uso de TLS y sus correspondientes certificados

PUESTA EN MARCHA DEL PROYECTO

Se despliegan tres redes:

- Red inicial sin TLS

Los archivos de configuración pueden encontrarse en el repositorio Github https://github.com/DFLBB/TFM_archs en la carpeta Arquitectura bajo el nombre TFM_sin_TLS.tar.

Se trata de una copia completa del proyecto, encontrándose los scripts utilizados en la carpeta scripts2 del repositorio github.

Se procede a desplegar la red y se conecta desde el navegador a la CouchDB y a Hyperledger Explorer.

The screenshot shows the Hyperledger Explorer interface. On the left, there's a sidebar with a tree view of the database structure, showing 'All Documents' under 'netchannel_peer'. The main area is a table with columns: ID, Nombre, docType, version, FechaAlta, FechaBaja, FechaDefinidaEnBlockchain, IPPeer, IPAmigo, IPPeerPuntoRaza, IDSess, and Nombre. The table lists multiple entries for 'PERROS' peers, each with a unique ID and specific details like creation and expiration dates, IP addresses, and session IDs. The dashboard at the bottom provides real-time metrics: 38 BLOCKS, 52 TRANSACTIONS, 21 NODES, and 12 CHAINCODES. It also includes a chart showing blocks per hour and transactions per hour over a 24-hour period, and a pie chart showing transactions by organization.

- Red con TLS



Los archivos de configuración pueden encontrarse en el repositorio Github https://github.com/DFLBB/TFM_archs en la carpeta Arquitectura bajo el nombre TFM_sin_TLS.tar.

Se trata de una copia completa del proyecto, encontrándose los scripts utilizados en la carpeta scripts del repositorio github.

Se modifica la red anterior para añadir TLS entre el CLI, el Orderer y los Peers. No se securizan las CA's ya que según indicaciones del tutor supone más problemas que ventajas.

- Red creada entre dos servidores

Los archivos de configuración pueden encontrarse en el repositorio Github https://github.com/DFLBB/TFM_archs en la carpeta Arquitectura bajo el nombre TFM_dos_Servidores.tar.

Se trata de una copia completa del proyecto, encontrándose los scripts utilizados en la carpeta scripts3 del repositorio github.

Se parte de la red con TLS para simular una situación más parecida a la realizada en la que cada servidor corresponde a una organización.

En primer lugar se crea una red Docker Swarm para conectar ambos servidores y posteriormente se modifican los archivos de configuración para dejar en cada servidor lo correspondiente a su organización así como los scripts para que lancen automáticamente la red entre ambos servidores.

Chaincodes y funcionalidades del sistema

CHAINCODES SEGÚN ORGANIZACIÓN

Los chaincodes utilizados por cada organización son los siguientes:

Federaciones Caninas

- Personas
- Perfil
- Perros
- Afíjos
- Solicitudes
- Veterinarios
- Microchip
- Razas
- Exposiciones y concursos
- Título

Colegios de Veterinarios

- Personas
- Perfil
- Perros
- Veterinarios
- Vacunas
- Microchips
- Razas



FUNCIONES COMUNES DE LOS CONTRATOS

Existen un conjunto de funciones que se definen dentro de los contratos. A modo de ejemplo mostraremos algunas:

- ejecutarConsulta: Permite realizar consultas al sistema a través del lenguaje de consulta de Mango, que se expresa como un objeto JSON que describe los filtros de los documentos de interés que se desean consultar.
- asignarEstado
- borrarEstado
- consultarEstado
- consultarRangoEstados
- getQueryResultForQueryString: Devuelve en formato JSON el resultado de una consulta, donde el TipoEstado tendrá la definición específica del estado consultado.

CHAINCODES

- PERSONAS

Contrato inteligente encargado de gestionar la identidad y sus datos personales de las personas que intervienen en las solicitudes que se lanzan contra el sistema. Actualmente gestiona los datos identificativos de:

- Criadores (propietarios de afijos)
- Propietarios de Perros
- Veterinarios
- Miembros de Federaciones

Está diseñado para admitir en el futuro ampliaciones de la blockchain, dando la posibilidad de albergar los datos de identificación de otro tipo de usuarios o datos asociados como jueces de certámenes caninos, miembros de Asociaciones caninas, miembros de Club de raza caninas y de grupos, miembros de laboratorios (para registros de ADN).

- AFIJOS

Contrato inteligente encargado de gestionar la identificación y propiedad de los afijos de criadores, denominación o marca personal registrada a través de una sociedad o federación canina que le autoriza a su titular o titulares a utilizarlo en la inscripción de camadas.

- PERROS

Contrato inteligente encargado de gestionar el registro de ejemplares canino, identificando su origen, pureza de raza, mencionando sus ascendentes y

descendientes. Identifica la propiedad del ejemplar y sus cambios , asi como la defuncion del ejemplar.



- **SOLICITUDES**

Importante contrato inteligente dentro de la blockchain, encargado de gestionar las solicitudes de los usuarios cuando es necesario realizar la validación o autorización de la acción por más de una persona, por ejemplo, cuando se desea vender un perro y existe mas de un propietario, o cuando se quiere certificar un cruce de dos perros pertenecientes a diferentes propietarios.

- **PERFILES**

Contrato inteligente encargado de gestionar los roles de los usuarios del sistema. Su uso está restringido exclusivamente a usuarios Administradores y permite definir perfiles de usuarios que permite el acceso a determinadas acciones y datos. Por ejemplo, solo un veterinario podrá poner un microchip o una vacuna.

- **VETERINARIOS**

Contrato inteligente encargado de gestionar los datos específicos de un veterinario como por ejemplo su número de colegiado.

- **MICROCHIPS**

Contrato inteligente encargado de registrar la identificación de microchip subcutáneo insertado por los veterinarios en el perro que contiene un transpondedor con un código único que permite identificar de manera univoca al ejemplar.

- **VACUNAS**

Contrato inteligente encargado de registrar el historial de las vacunas administradas por veterinarios y la duracion y tipo de proteccion de sus componentes.

- **RAZAS**

Contrato inteligente encargado de registrar gestiona el mantenimiento de los diferentes estándares de pureza de raza reconocidos, actualmente cerca de 346 razas, clasificadas en 10 grupos.

- **Otros posibles contratos inteligentes**

En un futuro podrían incorporarse con facilidad mas contratos a la Blockchain, como por ejemplo:

- **TÍTULOS:** Registra los títulos concedidos por federaciones, asociaciones y clubes por los méritos obtenidos por los ejemplares en exposiciones y/o concursos
- **EXPOSICIONES Y CONCURSOS:** Registra los resultados de los concursos y exposiciones de Morfología, Trabajo y disciplina, Rastreo y Agility que las diferentes federaciones, asociaciones y club realizan.
- **ADN:** Contrato que gestiona el registro de ADN. Facilitaría la certificación de autenticad de la linea genealógica y permitiría el estudio genético a partir de los datos obtenidos.
- **ENFERMEDADES / TRATAMIENTOS:** Contrato que gestiona el registro de las enfermedades y tratamientos que los veterinarios realizan sobre los perros, pudiéndose disponerse de un historial clínico para que cualquier veterinario pudiera consultarla cuando llegue un ejemplar a su consulta.

API-REST

Una vez desarrollada la parte de BlockChain, así como la parte de Big Data, se hace necesario presentar una interfaz gráfica que enlace estas dos tecnologías al usuario, de manera que la presentación sea más amigable.

Para este propósito se han utilizado las siguientes tecnologías:

- Capa de presentación
 - Entorno Web: HTML5, Jquery, JavaScript, CSS, Ajax, Bootstrap,
- Capa de enlace con el sistema (BlockChain y Big Data)
 - Api-Rest: Java con JAX-RS (servidor Tomcat)

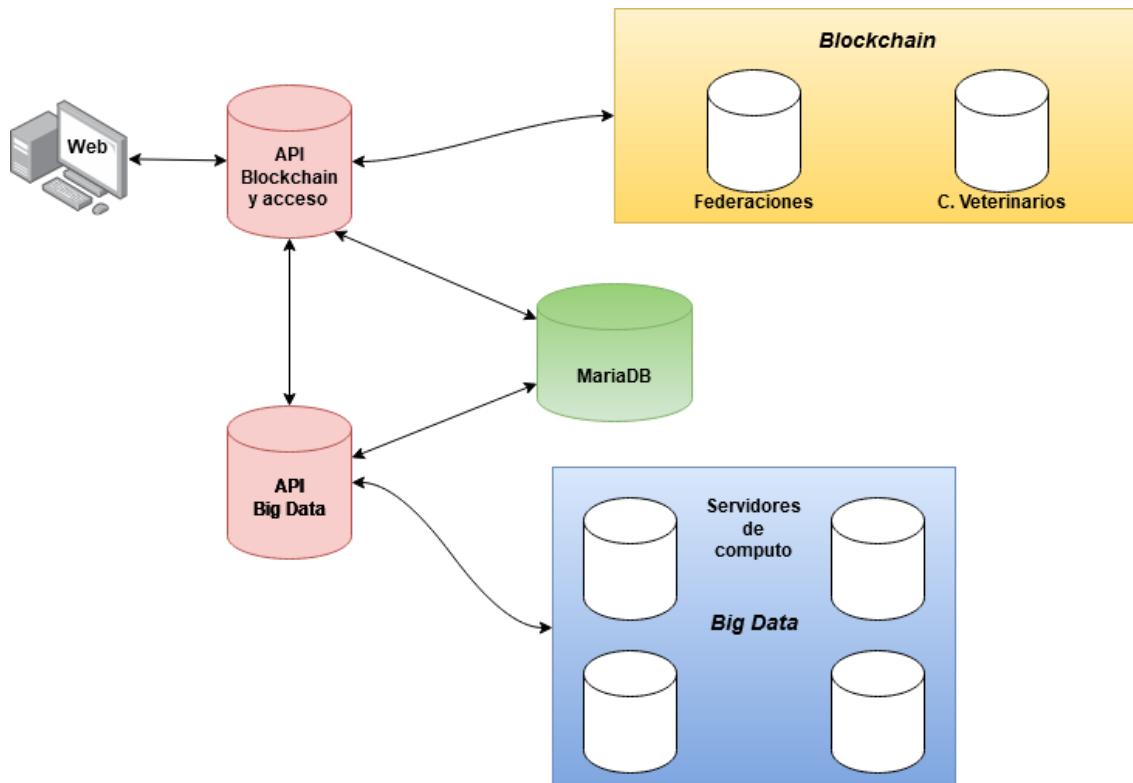
Un punto a tener en consideración es la existencia de una base de datos (*MariaDB*) conectada con la capa de presentación. El objetivo es reducir el tráfico de datos entre el entorno web y la BlockChain.

De esta manera, la recuperación de datos ya almacenados atacará contra la base de datos *MariaDB*, así como la conexión al sistema.

Cualquier operación relativa a altas y bajas en cualquier tipo de contrato (perros, usuarios, federaciones y veterinarios), accederá mediante api-rest a la BlockChain. En el *TFM_ANEXO_VI-API-REST_EntornoWeb.docx* se explica de una manera más detallada las tecnologías utilizadas.

A nivel de máquinas y arquitectura física desplegada tenemos lo siguiente:

- API BlockChain y base de datos MariaDb se encuentras en sendas maquinas en AWS, del tipo t2micro.
- Servidor BigData se encuentra en AWS en máquinas t3.2XLarge con procesadores GPU.
- Entorno BlockChain. Se utilizan dos servidores proporcionados por la UCM.



Desarrollo Big Data

Debido al gran impacto y desarrollo del concepto de Deep Learning en el campo de la inteligencia artificial, y a su posible aplicación con el mundo del blockchain canino, se ha decidido hacer uso de las técnicas más predominantes de este campo, para desarrollar un modelo de Red Neuronal Convolutacional para la clasificación de razas de perro. De manera que, el presente Trabajo de Fin de Máster está dotado de un extra en su implementación, para aumentar la funcionalidad de la página web, y favorecer al usuario que accede a la misma, la oportunidad de identificar cualquier raza de perro.

OBJETIVOS PRINCIPALES

Generar un modelo de Red Neuronal Convolutacional para la clasificación de razas de perro.

- Obtención de un dataset con más de 20000 imágenes de perros, y 120 razas distintas.
- Obtención del conjunto de datos de prueba y entrenamiento.
- Búsqueda de los mejores modelos preentrenados, y entrenamiento con distintos modelos.
- Comparación de los modelos entrenados y obtención del mejor modelo.
- Realización de las predicciones con los datos de prueba, y análisis de los resultados.
- Generar un script que permita la obtención de la predicción de la imagen que el usuario introduzca en la página web.

ANÁLISIS EXPLORATORIO DE LOS DATOS

Utilización del Standford Dog Dataset: 120 razas de perro, y 22.580 imágenes repartidas en 9960 destinadas al conjunto de prueba, y 10620 destinadas al conjunto de entrenamiento.



BigData 1. Razas de perro.

El dataset en cuestión es accesible desde el siguiente sitio Web:
(<https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>)

CONFIGURACIÓN DEL SERVIDOR AMAZON LINUX

Ha sido necesario el uso de un servidor de Amazon Web Service (AWS), con las siguientes características: 8 cores, 32 GB de RAM y 100 GB de memoria, correspondiente con el servidor de Amazon Linux Versión 33.0. De manera que, gracias al uso de este servidor, se han podido entrenar modelos distintos para una gran cantidad de datos.

ESTUDIO DE LOS MEJORES MODELOS CNNs

Con el objetivo de seleccionar el mejor modelo para el proyecto desarrollado, se han realizado entrenamientos con modelos diferentes, extraídos de los modelos preentrenados que nos pone a disposición la página oficial de Keras. (<https://keras.io/api/applications>). De manera que, evaluando la precisión de cada



uno de ellos, se han seleccionado los que se han considerado buenos modelos preentrenados: *VGG16*, *InceptionResNetV2*, y *MobileNetV2*.

DEFINICIÓN Y ENTRENAMIENTO DE LOS MODELOS

Es importante tener en cuenta que, los modelos que nos ofrece Keras ya están entrenados. De manera que, será necesario eliminar la última capa, realizada para un objetivo similar al propuesto, para poder reutilizar las capas ya entrenadas, e introducir funcionalidades adaptadas a nuestro objetivo principal con sólo añadir nuevas capas. Por tanto, para cada modelo seleccionado, se añaden las siguientes capas:

- **Capa añadida con el output de la última capa**, para la obtención promedio de las dimensiones espaciales. Para ello, se ha utilizado la función *GlobalAveragePooling2D*.
- **Capa con 512 nodos en la que se utilizará el rectificador lineal *activation relu***, que busca eliminar los valores negativos y dejar los positivos tal y como entran, es la función de activación más usada en deep learning y, especialmente, en los trabajos con imágenes.
- **Última capa densa, con 120 nodos**. Es la capa que identificará el tipo de raza de perro que tenemos en la imagen según los resultados que obtiene de las capas anteriores de la red. Esta capa utilizará la función *softmax*, que nos permite obtener la probabilidad de a qué clase de raza pertenece la imagen a identificar.

Por tanto, las capas anteriormente comentadas, serán las capas que se deberán de entrenar para cada modelo. De manera que, una vez que se ha definido la estructura que comprende cada modelo, se procede con el preentrenamiento y entrenamiento de las capas. Para ello, entran en juego el *ImageDataGenerator* y el *train_generator*, que se van a encargar del preentrenamiento, haciendo posible, tanto la lectura de la información de cada imagen, como el aumento del entrenamiento al conjunto train. Posteriormente, se compilan los modelos y se hace uso de la función *fit_generator* para proceder con el entrenamiento de los mismos.

COMPARACIÓN DE LOS MODELOS Y SELECCIÓN DEL MODELO GANADOR

A continuación, se muestra una tabla con la precisión obtenida por los diferentes modelos entrenados, únicamente se muestra la precisión obtenida en la etapa 20, es decir, la última precisión obtenida:

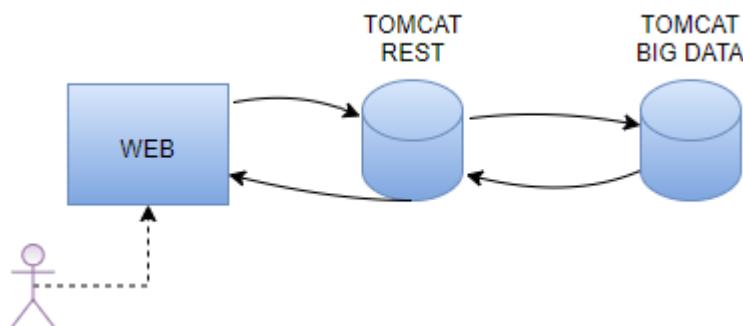
Modelo	Precisión
VGG16	0.87
InceptionResNetV2	0.87
MobileNetV2	0.92

BigData 2. Comparación de la precisión obtenida entre los modelos seleccionados

Se observa como claramente el modelo que utiliza la red preentrenada **MobileNetV2** obtiene una precisión superior a todos los demás modelos estudiados, por lo que elegimos este modelo como ganador.

GENERACIÓN DE UN APACHE TOMCAT EN EL SERVIDOR DE AMAZON LINUX.

Ha sido necesario el despliegue de un tomcat en el servidor de amazon Linux para conectar el Servicio REST con la parte de Big Data. De manera que, cuando un usuario introduzca una imagen en la página web, el servicio REST, será el encargado de devolver una ruta concreta de la imagen. De manera que, el servicio Rest tendrá que preguntar al tomcat de amazon linux de Big Data, por la URL en cuestión. Dicha URL será recibida por el modelo generado, y, en consecuencia, se generará una respuesta con el tipo de raza de perro correspondiente a dicha imagen. De manera que, la respuesta será enviada desde el tomcat de big data, hasta el servicio rest, y éste hará posible que el usuario visualice el tipo de raza de perro por la consola del sitio web implementado. Todo ello, se puede visualizar en la imagen

*BigData 2. Esquema sobre la estructura y conexión del servicio Rest y la aplicación Big Data*

PREDICCIONES CON LOS DATOS DE PRUEBA

Si analizamos la precisión del modelo ganador que se va a utilizar para la realización de las predicciones, se comprueba que es capaz de alcanzar aproximadamente el 0.9 de accuracy. Esto quiere decir que, cuando el usuario introduzca por el sitio web implementado la imagen del perro cuya raza quiera ser identificada, la probabilidad de acierto es elevada. A continuación, se muestra un ejemplo del resultado de la predicción en la web, cuando el usuario carga una imagen:



BigData 3. Resultados de la identificación de una raza de perro en la web.

CONCLUSIONES

En el presente Anexo V, sobre Big Data Canino, se ha desarrollado un modelo de red neuronal convolucional con una precisión de 0.92. Para ello, se han llevado a cabo los siguientes pasos, coincidentes con el cumplimiento de los objetivos iniciales:

- Se ha utilizado un dataset de 22.580 imágenes, llamado Standford Dog Dataset. Obtenido del siguiente sitio web:
<https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>.
- Obtención del conjunto de datos de prueba y entrenamiento. Para ello, se ha desarrollado un script que introduce 83 imágenes de cada tipo de raza en una carpeta llamada Images_Test. Por lo tanto, finalmente se tienen 9960 imágenes destinadas al conjunto de prueba, y 10620, destinadas al conjunto de entrenamiento.
- Búsqueda de los mejores modelos preentrenados, y entrenamiento con distintos modelos. Se han evaluado los modelos presentes en la página oficial de Keras, y se han entrenado en un servidor de Amazon de pago para poder compararlos y seleccionar el mejor de ellos. Los modelos seleccionados han sido: VGG16, InceptionResNetV2 y MobileNetV2.
- Comparación de los modelos entrenados y obtención del mejor modelo. Se han comparado los tres modelos seleccionados y se ha seleccionado el modelo mobilenet, como mejor modelo, por tener mayor precisión.
- Realización de las predicciones con los datos de prueba, y análisis de los resultados.
- Generar un script que permita la obtención de la predicción de la imagen que el usuario introduzca en la página web.



VERSION
FINAL

15-9-2020

Blockchain & BigData Canino

ANEXO I: Introducción al Mundo Canino



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



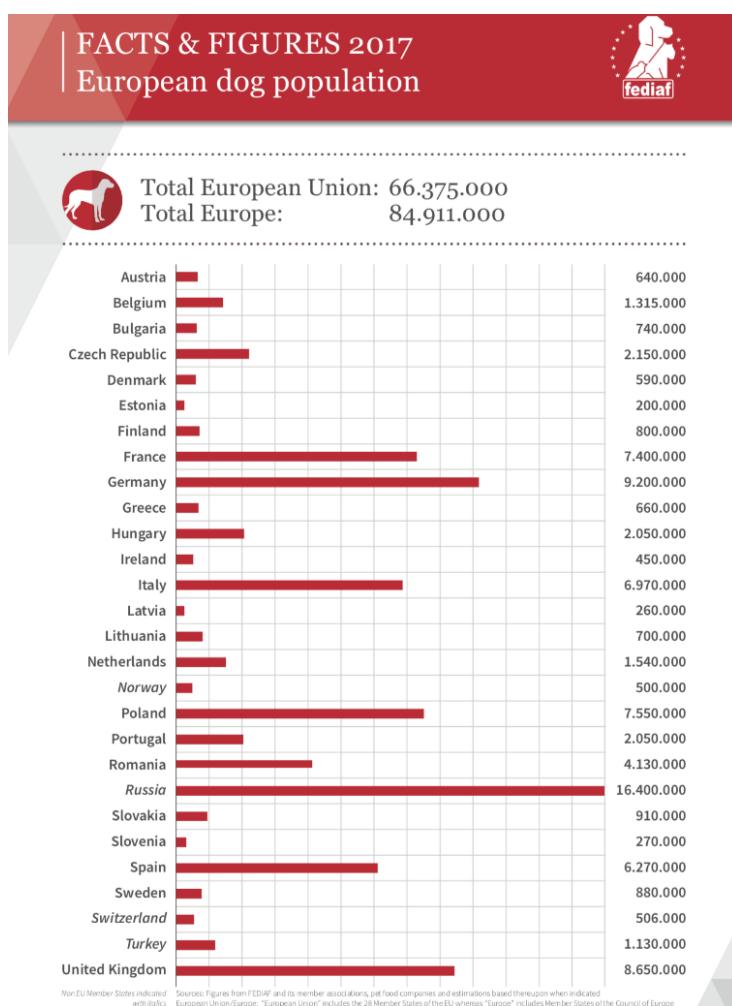
ÍNDICE

El negocio multimillonario de las mascotas	2
Conceptos básicos del mundo canino.....	5
IDENTIFICACION CANINA	5
LIBRO GENEALOGICO.....	6
FEDERACIONES CANINAS.....	7
SOCIEDADES y CLUBES DE RAZA.....	7
AFIJOS DE CRIADORES	8
CERTIFICACIONES.....	10
REGISTRO DE CAMADAS Y PROPIETARIOS.....	12
Blockchain Canino.....	15
FUNCIONALIDADES BÁSICAS A DESARROLLAR	15
AMPLIACIONES POSIBLES	15
Big Data Canino	16
AMPLIACIONES POSIBLES	16
Conclusiones.....	17
EL MUNDO CANINO Y EL BLOCKCHAIN	17

El negocio multimillonario de las mascotas

En la actualidad el mundo de las mascotas constituye un negocio multimillonario de aproximadamente 36.500 millones al año en toda Europa, donde hay 200 millones de mascotas en 80 millones de hogares. En España genera más de 1.000 millones anuales de negocio que generan más de 100.000 puestos de trabajo, entre indirectos y directos.

Según datos de la Red Española de Identificación de Animales de Compañía (REIAC) existen en el país más 13 millones de mascotas registradas, de las cuales más de la mitad corresponden a ejemplares caninos.

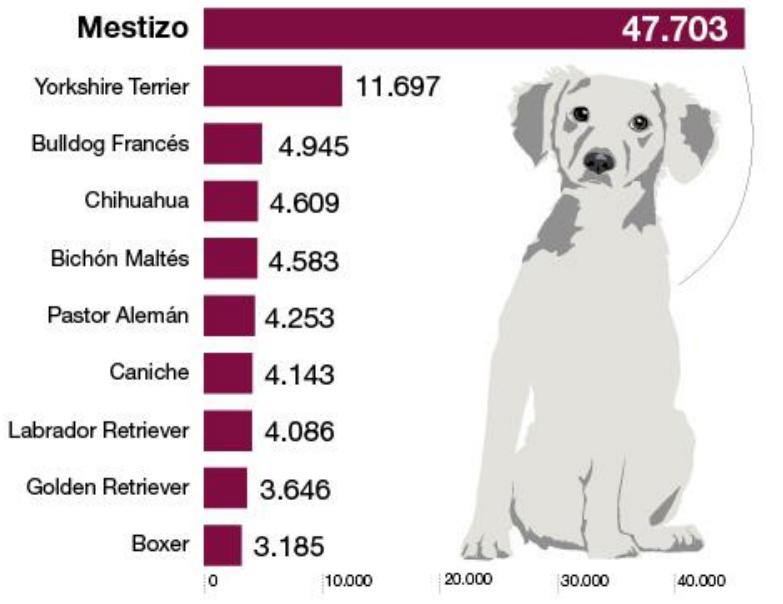


Los ejemplares de raza que actualmente reciben más número de registros son los de la raza Yorkshire Terrier con más de 11.000 inscripciones en el último año, según AIAC.

El precio de venta de un cachorro varía según el sexo del animal, el criador y las circunstancias del momento, pero rondaría los 1.500-2500€ por cachorro de

media (no obstante, existen casos variados en los que se encuentren por 400€ o por 60.000€ en el caso de un campeón del mundo)

RAZAS MÁS COMUNES



FUENTE: AIAC

EL PERIÓDICO

La pureza de raza, la línea genealógica y los premios que han obtenido sus ancestros determinan mucho el precio, por lo que la verificación y certeza de estos datos son especialmente relevantes en este tipo de negocio donde existen muchos casos de personas que se aprovechan de la gente.



Actualmente existen multitud de organizaciones en diferentes países que gestionan las certificaciones del mundo canino, cada una encargándose de

diferentes partes procesos (registros de afijos de criadores, camadas, propietarios, identificación canina, premios y títulos, ...) y en muchos casos de manera repetida, duplicando competencias, por no existir un registro unificado.

Con el fin de poder reunir a estos colectivos en un solo entorno compartido se plantea un desarrollo Blockchain que sirva para interactuar inicialmente entre:

- Criadores caninos
- Propietarios
- Veterinarios
- Federaciones caninas
- Jueces
- Sociedades y Clubes caninos



Conceptos básicos del mundo canino

IDENTIFICACION CANINA

Antiguamente los ejemplares caninos llevaban consigo un tatuaje a modo de documento de identidad. En la actualidad este proceso lo realizan los veterinarios insertando un pequeño microchip subcutáneo del tamaño de un grano de arroz, en una capsula de cristal especial que contiene un transpondedor con un código único.



La implantación de este microchip es obligatoria en España con el objeto de evitar el abandono animal estando tipificado como delito en la [Ley orgánica 1/2015](#).

Existe un registro gestionado por cada comunidad autónoma así como una Red Española de Identificación de Animales de Compañía (REIAC), que en muchos casos no está actualizado.





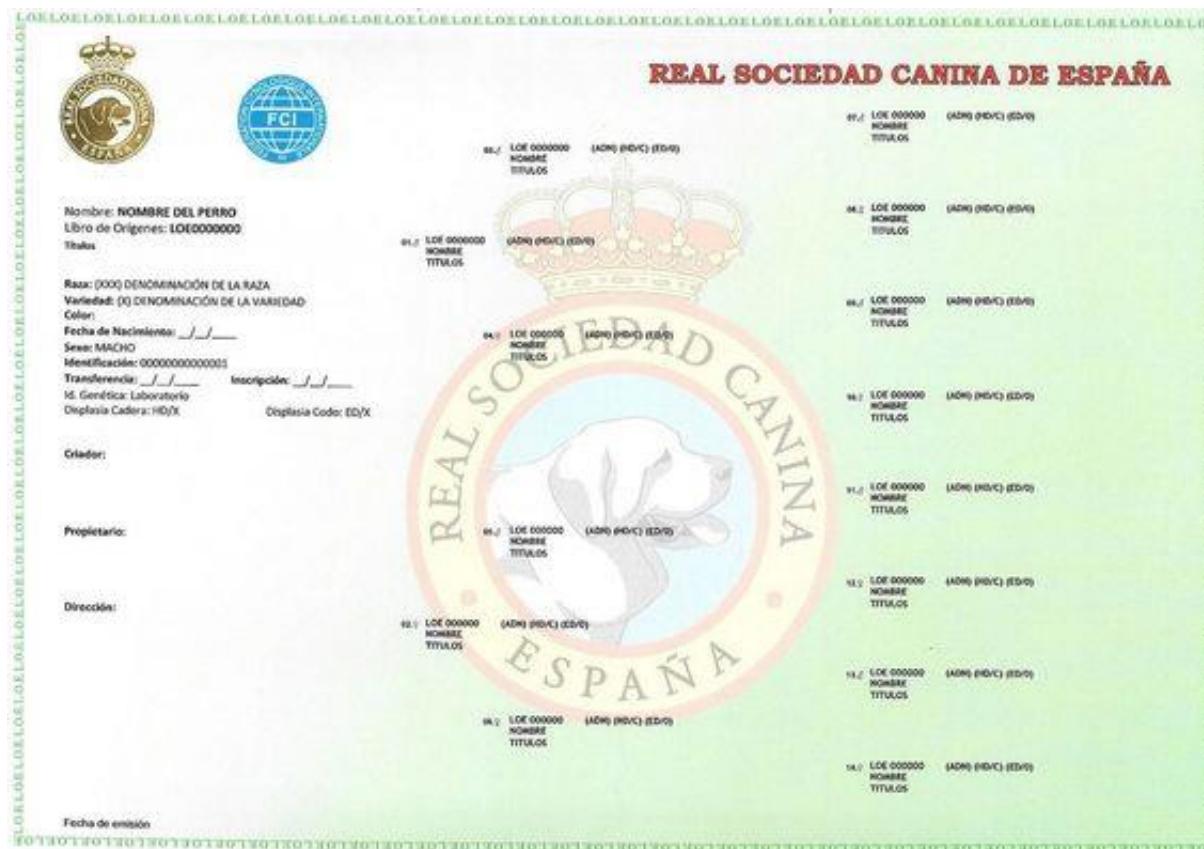
LIBRO GENEALOGICO

Registro, fichero o sistema informático donde se inscribe los perros de raza pura, mencionando sus ascendentes y descendientes, gestionado por organizaciones reconocidas oficialmente a través de eventos.

El hecho de afirmar que un perro tiene pedigree indica que ejemplar está inscrito en un [libro genealógico](#) reconocido por una organización oficial.

Para su inscripción es necesario demostrar que las tres últimas generaciones del ejemplar se encuentran inscritas en un libro de origen de especies.

En caso de que este hecho no pudiera ser demostrado, existe la posibilidad de realizar un reconocimiento de raza por un juez especialista de la organización, si cumple con todas las estrictas [características de estándar](#) establecidas para la raza.





FEDERACIONES CANINAS

Estas organizaciones se aseguran de que los pedigüeños, los criadores y los jueces sean reconocidos mutuamente por todos los miembros que las integran. Organizan [concursos y exposiciones](#) de Morfología, Trabajo y disciplina, Rastreo y Agility, concediendo menciones y títulos a los mejores ejemplares, a través de jueces reconocidos por dichas organizaciones.

Existen numerosas federaciones a nivel nacional e internacional, por ejemplo:

- [Fédération Cynologique Internationale \(FCI\)](#)
- [Real Sociedad Canina Española \(RSCE\)](#)
- [The Kennel Club \(TKC\)](#)
- [Alianz canine Worldwide \(ACW\)](#)
- [American Kennel Club \(AKC\)](#)
- [European Guide Dog Federation](#)
- [Federación Cinológica Española \(FCE\)](#)
- [Canina nacional \(ACCAM\)](#)

SOCIEDADES y CLUBES DE RAZA

Son organizaciones que colaboran con las federaciones para realizar diferentes gestiones a nivel local para el fomento de razas caninas o de alguna en especial. Preparan en coordinación con las federaciones concursos y exposiciones, por ejemplo:

- [Sociedad Canina de Bizkaia](#)
- [Sociedad Canina Montañesa](#)
- [Sociedad valenciana para el fomento de razas caninas](#)
- [Sociedad Canina Gallega](#)
- [Sociedad Canina de Murcia](#)
- [Sociedad Canina de Andalucía Occidental](#)
- [Club Español del Yorkshire Terrier \(CEYT\)](#)
- [Asociación nacional para el fomento del Bulldog Francés](#)
- [Asociación española de perros nórdicos y akita](#)
- [Asociación de criadores de perros de pastor alemán](#)
- [Club Collie de España](#)



AFIJOS DE CRIADORES

Un afijo es una denominación o marca personal de un criador que lo registra a través de una sociedad o federación canina.

La concesión de un afijo autoriza a su titular o titulares a utilizarlo en la inscripción de camadas. Todos los titulares del afijo deben ser propietarios de la hembra, madre de la camada, para poder utilizar el afijo.



El afijo es propiedad exclusiva de la persona o colectividad que ha adquirido el derecho y es vitalicio.

El propietario de un afijo puede aplicarlo a todos los ejemplares de cualquier raza de la que es criador. No puede aplicarse a un ejemplar un afijo diferente al de su criador. Nadie podrá ser titular de más de un afijo por raza.





Por ejemplo:

- Afijo: [Coramonte](#)
- Ejemplares: [Vanessa de Coramonte](#), [Gastón de Coramonte](#),

Jr. Ch. Quimera de Coramonte  Nacido: 03/05/2015 Propietario: Elena Soler Criador: Jesus Montero	Sofia De Coramonte  Nacido: 20/05/2015 Propietario: Pilar Fernandez Criador: Jesus Montero	Gaston De Coramonte  Nacido: 08/07/2010 Propietario: Jesus Montero Perdigero Criador: Jesus Montero Perdigero
Dulce De Coramonte  Propietario: Jesus Montero Perdigero Criador: Jesus Montero Perdigero	Caramelo de Coramonte  Nacido: 05/05/2014 Propietario: Jesus Montero Criador: Jesus Montero	Ch. Vanesa de Coramonte  Propietario: Unai Ares Criador: Jesus Montero

La organización que registra el afijo extiende un certificado para su propietario o propietarios.

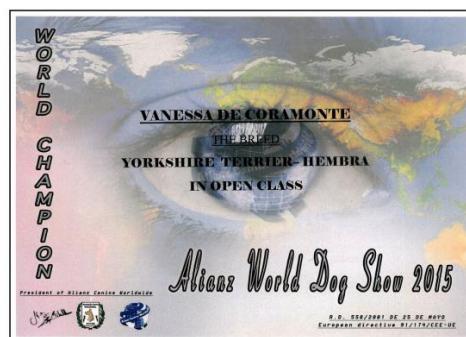




CERTIFICACIONES

En los apartados anteriores hemos mencionado ya algunos de las certificaciones que debería proporcionar el proyecto blockchain a desarrollar, por ejemplo:

- Certificado de afijo de criador
- Certificado de identificación canina (microchip)
- Certificado del pedigree del ejemplar
- Certificado de reconocimiento de raza del ejemplar
- Certificado premios y títulos del ejemplar





Sin embargo, es necesario estudiar el ciclo completo de registro de los perros desde el momento del nacimiento, que lo desarrollaremos en el apartado siguiente, y observaremos que requiere algunas certificaciones adicionales a las anteriormente citadas, por ejemplo:

- Certificado de propietario de ejemplar
- Certificado de nacimiento de camadas
- Certificado de traspaso de propiedad
- Certificado de cesión temporal de hembra





REGISTRO DE CAMADAS Y PROPIETARIOS

Los actuales procedimientos de las federaciones caninas se realizan mayoritariamente de manera presencial (ya que es necesario realizar pagos por las gestiones), lo que supone desplazarse hasta sus oficinas.

Primeramente, un criador debe registrar su marca personal o afijo, para que los cachorros que críe tengan su apellido.

En segundo lugar, para poder inscribir el nacimiento de una camada, la propiedad de sus progenitores debe estar registrados bajo su nombre.



En algunos casos los criadores realizan cruces de un ejemplar hembra de su propiedad con un macho de otro propietario. Este debe certificar por escrito por ambas partes.

Los servicios de monta o reproducción suelen estar regulados por un contrato privado entre ambas partes, que aseguran la veracidad del cruce y del pago por el servicio, reciben compensaciones económicas y/o algún cachorro del resultado del cruce.

Esta documentación es necesario aportarla antes de transcurrir un mes del nacimiento para que la [notificación de una camada](#) sea registrada, junto con el nombre y sexo de los cachorros. El conste de la gestión será en función del número de cachorros que tenga la camada.



	REAL SOCIEDAD CANINA DE ESPAÑA
Declarada de Interés Público por Real Orden del Ministerio del Interior, Decreto 27/02/1918.	
Miembro de la Federación Cynologique Internationale (FCI) desde 1912.	
C/Lagasca, 16 - 28003 Madrid - Tfno. 91 426 49 60 - Fax 91 425 11 13 - www.rscce.es	

Tras el pago de las tasas de gestión la federación proporcionara al criador un justificante de ingreso provisional de cada cachorro que asevera su propiedad, teniendo un plazo máximo de 6 meses para que esta documentación sea devuelta cumplimentando la identificación del microchip que le ha sido insertado y la firma del veterinario que realizó la operación y registro de propiedad (*existe algunas excepciones de plazo que ahora no vamos a explicar por simplificar la explicación del proceso*)

Una vez finalizada esta gestión la federación entregará al propietario un certificado de registro o por un coste adicional un [certificado del pedigree del perro](#).



Estos documentos servirán para realizar un cambio de propietario, que será necesario registrarla en la federación, tras una nueva tasa de gestión, y poder recibir un certificado o pedigrí con los datos actualizados.

El nombre del afijo que constará en el registro será el del propietario del ejemplar hembra. Es por ello que existe un trámite para la cesión temporal de una hembra a otro criador.



Blockchain Canino

FUNCIONALIDADES BÁSICAS A DESARROLLAR

Aunque la lista de requisitos de funcionalidades será necesario desarrollar cuando iniciemos el TFM, podríamos ir adelantando algunas que parecen inevitables por el momento:

- Identificación y autenticación (Criadores caninos, propietarios, veterinarios, sociedades caninas, jueces, clubes caninos)
- Afijo de criador (alta, baja, modificación, certificación)
- Camadas (registro y certificación)
- Ejemplar (traspaso de propiedad, certificación propiedad, pedigree y defunción)
- Microchip (registro y modificación)
- Registro de vacunación
- Concursos y exposiciones (registro de premios y títulos)



AMPLIACIONES POSIBLES

Existen algunas ideas sobre las ampliaciones que se podrían realizar dependiendo de la complejidad de implementación:

- Identificación a través de ADN
- Registro de enfermedades



Big Data Canino

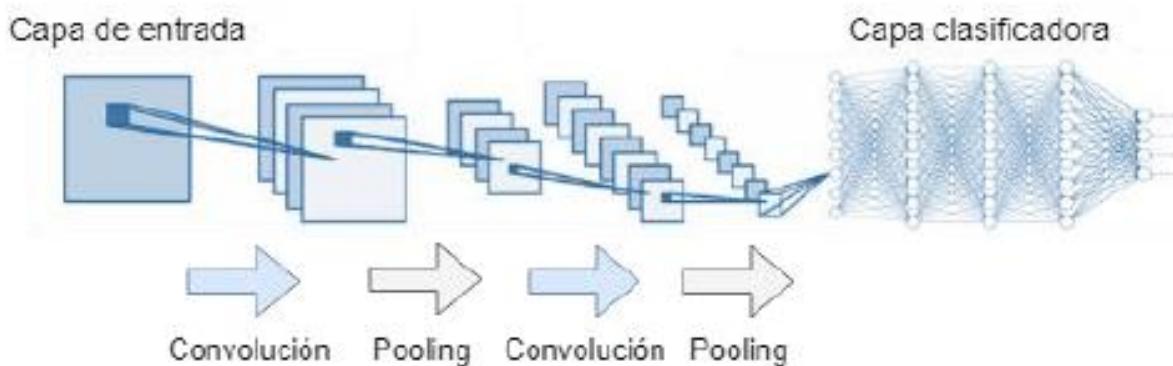
AMPLIACIONES POSIBLES

Existen algunas ideas sobre las ampliaciones que se podrían realizar a partir de la Base de Datos ya generada con Blockchain y nuevos datos acumulados:

- Calidad estimada del cruce de razas a partir de los modelos obtenidos por las líneas genealógicas y los premios y títulos conseguidos.
- Estudio de características morfológicas a partir del ADN y detección de enfermedades

Para esta prueba de concepto se ha optado por la implementación de :

- Reconocimiento de razas a través de imágenes (Deep learning)





Conclusiones

EL MUNDO CANINO Y EL BLOCKCHAIN

En estas reducidas líneas hemos intentado dar unas primeras pinceladas de la complejidad que se esconde detrás del mundo canino donde existen múltiples participantes con intereses en algunas ocasiones en común y en otros particulares sobre todo en cuanto comprendemos que detrás de las líneas genealógicas de los perros., el prestigio, los concursos y exposiciones se mueve mucho dinero.

Quizás sea esta la razón por la cual cada organización se muestre recelosa, para compartir su información, produciéndose en algunos casos perdida de información, datos duplicados o no incongruentes por falta de actualización.

Por otro lado, existen los amigos de lo ajeno que intentan engañar y aprovecharse de personas, falseando datos y cometiendo fraudes, en contraposición con algunas asociaciones cuyo fin es el fomento de las razas caninas, pero se ven limitados por no disponer de los recursos necesarios para abordar un proyecto global.

Es por ello, que un blockchain canino podría resolver muchos de los problemas citados.





VERSION
FINAL

15-9-2020

Blockchain & BigData Canino

ANEXO II: Arquitectura Blockchain



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Documentación del TFM
Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE

Tecnología a utilizar	2
Arquitectura del proyecto	4
INSTALACIÓN DE HYPERLEDGER FABRIC	6
ORGANIZACIÓN DE LA RED HYPERLEDGER FABRIC.....	8
ESTRUCTURA DEL PROYECTO	9
SCRIPTS PARA LEVANTAR EL PROYECTO	12
CONFIGURACIÓN DE LA RED	21
Puesta en marcha del proyecto	33
RED INICIAL SIN TLS	33
INSTALACIÓN DE HYPERLEDGER EXPLORER.....	42
RED CON TLS	49
RED CREADA ENTRE DOS SERVIDORES	54
• Creación de la red Docker Swarm.....	54
• Configuración de la red	55
Bibliografía	76



Tecnología a utilizar

Una vez definido el caso de uso, consistente en una red entre las organizaciones Federaciones Caninas y Colegios Veterinarios para facilitar la realización de las distintas acciones que los propietarios de perros tienen que llevar a cabo en cada una de estas organizaciones y evitar los posibles casos de fraude que puedan intentar cometerse, se procede a estudiar cuál será la mejor tecnología y arquitectura para generar la red entre ambas.

En primer lugar se decide el uso de la tecnología Blockchain para regir las relaciones entre ambas organizaciones, lo que redundaría en que la actividad sea transparente para ambas y en que la información sea inmutable y no pueda ser modificada para realizar ningún tipo de fraude.

Por otra parte, dado que se trata de dos organizaciones privadas que deben interactuar sin llegar a tener confianza una en la otra y que los datos tienen una componente privada (siendo algunos de ellos sensibles y protegidos por el RGPD), se propone la utilización de una red Blockchain privada con la tecnología Hyperledger Fabric como base.

Los activos con los que contará la red son los perros, que tendrán un propietario que podrá cambiar, se les podrá poner el microchip y vacunas, participarán en concursos,...

Los participantes serán las Federaciones Caninas y los Colegios Veterinarios, que son los que ejecutarán las acciones sobre los perros.

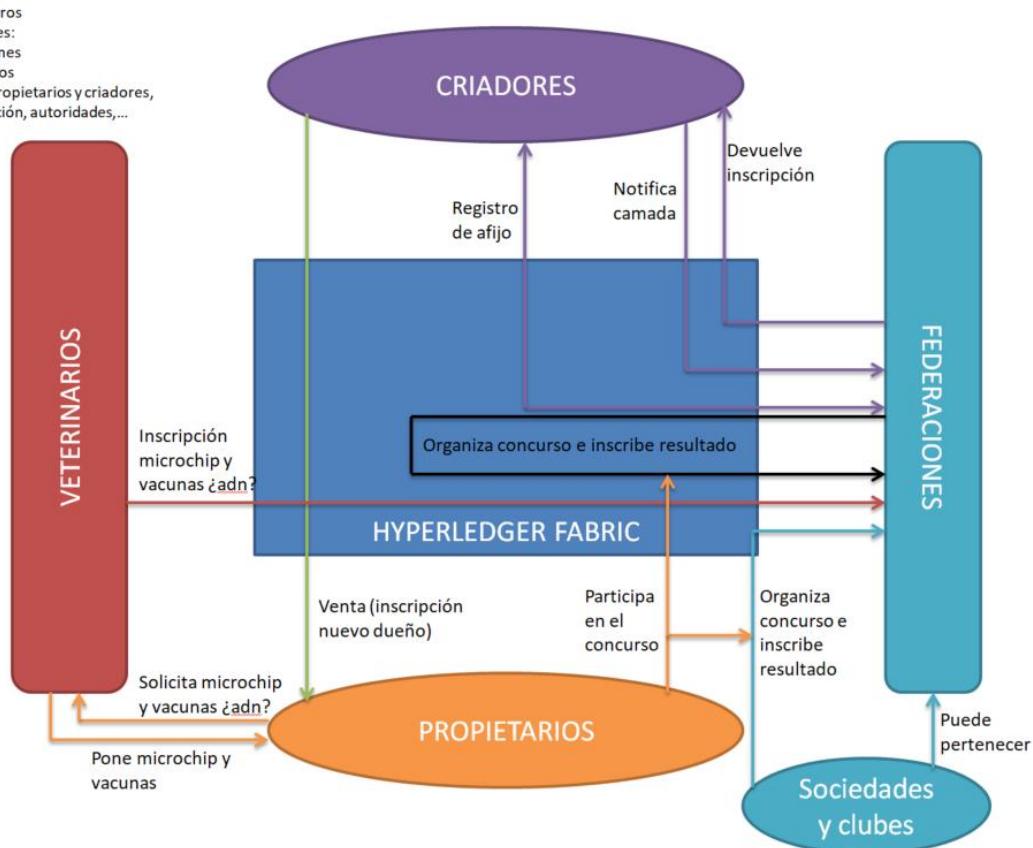
Por otra parte que los usuarios serán las propias Federaciones, los Colegios Veterinarios, los propietarios, los veterinarios y las autoridades competentes (agentes de la autoridad, juzgados,...) a los que se permita el uso de la red.

En el siguiente gráfico se describen las relaciones entre los participantes en la red, que se gestionan a través de Hyperledger Fabric:

- Criadores: Solicitarán el afijo a la Federación correspondiente y notificarán la camada a la Federación, recibiendo la inscripción de la camada
- Veterinarios: Colocarán e inscribirán el microchip y las vacunas en la Federación. Tendrán relación con los propietarios, que solicitarán estas acciones.

- Como propuesta a futuro los veterinarios tomarán una muestra de ADN que quedará registrada en la Blockchain y además podría utilizarse para realizar un algoritmo que permita determinar los mejores cruces entre los perros para obtener los mejores ejemplares
-
- Federaciones: Inscribirán los afijos, las camadas, los microchips y vacunas, la propiedad de los perros, los resultados de los concursos y facilitarán el pedigree a los propietarios que lo soliciten. Además podrá haber otros clubes y asociaciones que realicen concursos, pertenezcan o no a la Federación.
-
- Propietarios: Solicitarán la inscripción de la propiedad del perro a la Federación, el microchip y vacunas a los veterinarios, presentarán perros a concursos, solicitarán pedigríes,...

Descripción:
 Activos: Perros
 Participantes:
 - Federaciones
 - Veterinarios
 Usuarios: Propietarios y criadores,
 administración, autoridades,...





Arquitectura del proyecto

Dentro de Hyperledger Fabric se definen dos organizaciones con las siguientes características:

- **Federaciones Caninas**

Se trata de las organizaciones que llevan a cabo el control de los afijos autorizados para la cría de perros, control de las características de las razas, pedigreees,...

En este caso se tendrán en cuenta las cuatro mayores Federaciones existentes:

- FCI: Fédération Cynologique internationale
- RSCE: Real Sociedad Canina Española
- TKC: The Kennel Club
- ACW: Alianz Canine Worldwide

- **Colegios Veterinarios**

Se trata de las organizaciones que llevan el control de todo lo relativo a la sanidad de los perros: colocación de microchip, vacunas, pasaporte sanitario,...

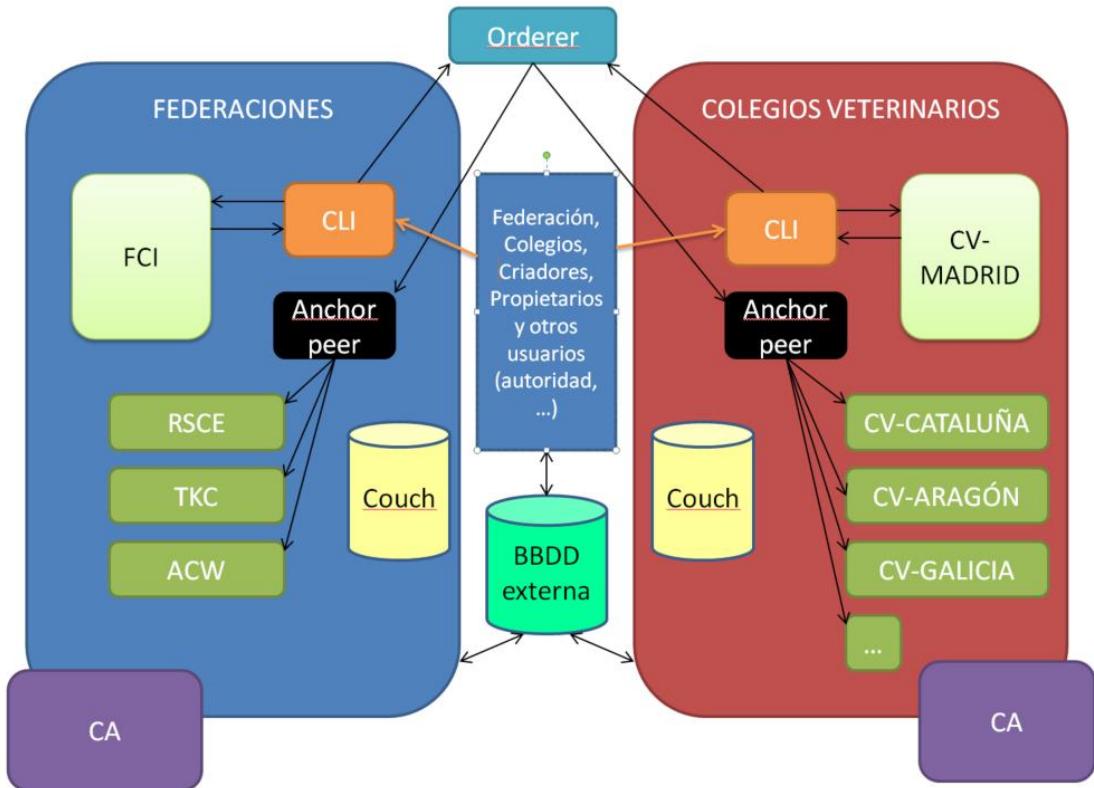
Se tomarán los Colegios Veterinarios a nivel de Comunidad Autónoma, por lo que tendremos diecisiete Colegios Veterinarios

Se utilizará CouchDB en vez de levelDB para aumentar las posibilidades de consulta a la red y cada organización tendrá una autoridad de certificación (CA).

Además se implementará una base de datos externa para guardar los datos sensibles y los destinados al uso por la aplicación de Big Data.

El tipo de consenso será “solo” para el desarrollo, aunque posteriormente debería implementarse “Kafka” en producción.

Una vez definidas las organizaciones la arquitectura será la siguiente:



Cada organización dispondrá de un nodo para cada uno de sus componentes, existiendo además un CLI al que atacarán los usuarios desde la API, un Orderer que distribuirá las transacciones por los nodos de la red, un Anchor Peer al que el Orderer enviará la información, una CouchDB en cada peer, una CA (autoridad de certificación) para cada organización y una base de datos externa en la que se guardarán los datos sensibles y los destinados al uso por la aplicación de Big Data.



INSTALACIÓN DE HYPERLEDGER FABRIC

Para la instalación se utilizará un usuario al que daremos de alta con el nombre “hyperledger”.

Una vez instalados los prerequisitos de hyperledger para Ubuntu que se encuentran en:

<https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh>

Y que incluyen docker y node.js, tras instalarlos se procede a instalar docker-compose desde la dirección:

[https://github.com/docker/compose/releases/download/1.21.2/docker-compose-\\$\(uname -s\)-\\$\(uname -m\)](https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m))

Y por último go:

<https://dl.google.com/go/go1.13.3.linux-amd64.tar.gz>

Las instrucciones concretas para realizarlo han sido:

```
# Añadir usuario hyperledger
adduser hyperledger

# Dar permisos root al usuario
sudo usermod -a -G sudo hyperledger
sudo vi /etc/sudoers

# Añadir bajo el apartado User privilege specification: hyperledger  ALL=(ALL:ALL) ALL
logout

# Entrar al servidor con el usuario hyperledger

# Descargar los prerequisitos
curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh

# Dar permisos de ejecución
chmod u+x prereqs-ubuntu.sh

# Ejecutar los prerequisitos
./prereqs-ubuntu.sh

# Actualizar docker-compose y dar permisos de ejecución
sudo curl -L https://github.com/docker/compose/releases/download/1.21.2/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
# Actualizar
sudo apt-get update
```



```
sudo apt-get -y upgrade

#/ Instalar GO
sudo curl -O https://dl.google.com/go/go1.13.3.linux-amd64.tar.gz
sudo tar -xvf go1.13.3.linux-amd64.tar.gz

#/ Editar el archivo bashrc y añadir al final del fichero las siguientes variables de estado
sudo vi ~/.bashrc

export GOROOT=$HOME/go
export GOPATH=$HOME/work
export PATH=$PATH:$GOROOT/bin:$GOPATH/bin

source ~/.bashrc

logout

#/ Comprobar que se ha instalado GO
go version
```

Una vez preparado el servidor se procede a descargar e instalar Hyperledger Fabric.

Instalaremos Fabric bajo el home del usuario hyperledger con la siguiente ruta:

/home/hyperledger/work/src

Las instrucciones serán:

```
# Creamos la ruta a utilizar
mkdir work && cd work
mkdir src && cd src

# Descargamos el repositorio y los binarios de Fabric utilizando las versiones correctas
git clone -b master https://github.com/hyperledger/fabric-samples.git

cd ~/work/src/fabric-samples
git checkout v1.4.0
curl -sSL http://bit.ly/2ysbOFE | bash -s -- 1.4.0 1.4.0

# Comprobamos la ruta para exportar la variable de estado PATH
pwd

export PATH=/home/hyperledger/work/src/fabric-samples/bin:$PATH
```

En este punto el servidor ya está preparado para utilizar Hyperledger Fabric.



ORGANIZACIÓN DE LA RED HYPERLEDGER FABRIC

El proyecto lo situaremos bajo la carpeta genérica que ha creado hyperledger fabric durante la instalación en:

/home/hyperledger/work/src/fabric-samples

Donde crearemos una carpeta llamada TFM, siendo la ruta del proyecto:

/home/hyperledger/work/src/fabric-samples/TFM

En primer lugar crearemos una red con la arquitectura descrita anteriormente:

- Dos organizaciones: Federaciones y ColegiosVeterinarios
- Cuatro peers para la primera organización y diecisiete para la segunda
- Orderer
- CLI
- CA's
- CouchDB

En la que instalaremos Hyperledger Explorer para poder visualizar la información de la red en el navegador, accediendo también a la CouchDB desde el navegador.

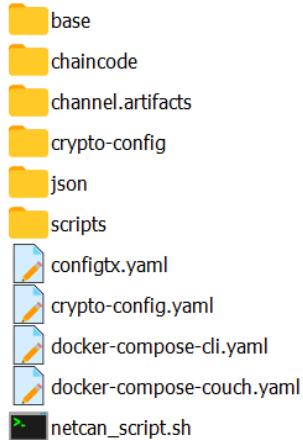
Posteriormente añadiremos TLS a la red, generando una red más segura en la que será necesario proveer los certificados correspondientes al efectuar cualquier acción.

Por último desplegaremos la red entre dos servidores simulando la existencia real de las dos organizaciones.

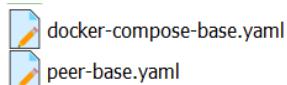


ESTRUCTURA DEL PROYECTO

Para la realización del proyecto, bajo la carpeta del proyecto /TFM, se generarán:



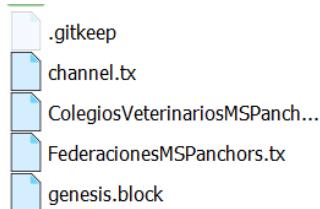
- Los archivos de configuración del proyecto de fabric: configtx.yaml, crypto-config.yaml, docker-compose-cli.yaml y docker-compose-couch.yaml



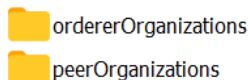
- La carpeta base en donde se situarán dos archivos de configuración de los que extienden los dos archivos docker-compose anteriores: docker-compose-base.yaml y peer-base.yaml



- La carpeta chaincode, donde se situarán los chaincodes desarrollados para la red: afijos, microchips, perfiles, perros, personas, razas, solicitudes, vacunas y veterinarios, y una carpeta netcan con datos comunes a todos ellos



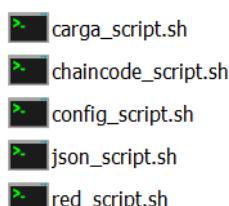
- La carpeta channel-artifacts, donde se generarán los artefactos del proyecto



- La carpeta crypto-config, donde se generará el material criptográfico



- La carpeta json, donde colocaremos los archivos json generados con los datos iniciales para cargar a la Blockchain: los afijos existentes, perros con sus dependencias (padres, hijos,...), propietarios, veterinarios, microchips, vacunas,...





- La carpeta scripts, donde colocaremos una serie de scripts para levantar el proyecto de manera automática, de manera que se levante la red, se configure, se instalen e instancien los chaincodes desarrollados y se carguen los datos iniciales

```
netcan_script.sh  
stop_netcan_script.sh
```

- Por último dos scripts para lanzar el proyecto y pararlo y limpiar del servidor los dockers y chaincodes levantados
- Al ejecutar el script netcan_script.sh lo primero que hace es acceder a un repositorio en github creado para el proyecto donde se encuentran actualizados el resto de scripts, los archivos json de carga inicial actualizados y los chaincodes actualizados, descargando dichas carpetas al servidor y sustituyendo las existentes (si las hay), para que el proyecto esté siempre actualizado.

Dicho repositorio puede encontrarse en:

https://github.com/DFLBB/TFM_archs

The screenshot shows the GitHub repository page for 'DFLBB / TFM_archs'. The repository has 1 branch and 0 tags. The master branch contains the following files and folders:

- Arquitectura
- chaincode/netcan
- documentacion
- json
- script_mariadb_carga_inicial
- scripts1
- scripts2
- scripts3

SCRIPTS PARA LEVANTAR EL PROYECTO

- netcan_script.sh

Es el único script que hay que lanzar para inicializar el proyecto.

En primer lugar limpia la instalación, a continuación accede al repositorio de Github, descarga las carpetas json, chaincode y scripts y sustituye las existentes por estas y posteriormente va lanzando el resto de scripts de configuración, ya sea para uno o dos servidores

- red script.sh

Este script levanta la red hyperledger Fabric y copia los scripts necesarios al docker del CLI, donde luego habrá que realizar la configuración del proyecto

- Serv2_script.sh

Este script realiza la configuración en el servidor 2 en el caso de que se levante la red entre dos servidores

```
docker stop $(docker ps -a -q)
docker rm $(docker ps -a -q)
docker volume prune --force
docker system prune --force
docker network prune --force

echo "*****"
echo "Levantando la red en el servidor ColegiosVeterinarios"
echo "*****"
docker-compose -f docker-compose-cli.yaml -f docker-compose-couch.yaml up -d

""
```

- config_script.sh

Este script se ejecuta dentro del docker del CLI y realiza la configuración del canal, los peers y los pares de anclaje, ya sea con TLS o sin TLS



```

peerOrganizations/colegiosveterinarios.netcan.com/peers/cvmurcia.colegiosveterinarios.netcan.com/tls/ca.crt peer channel join
-b netcanchannel.block
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/colegiosveterinarios.netca
n.com/users/Admin@colegiosveterinarios.netcan.com/msp/ CORE_PEER_ADDRESS=cvnavarra.colegiosveterinarios.netcan.com:7051 CORE_P
EER_LOCALMSPID="ColegiosVeterinariosMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
/peerOrganizations/colegiosveterinarios.netcan.com/peers/cvnavarra.colegiosveterinarios.netcan.com/tls/ca.crt peer channel join
-b netcanchannel.block
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/colegiosveterinarios.netca
n.com/users/Admin@colegiosveterinarios.netcan.com/msp/ CORE_PEER_ADDRESS=cvpaisvasco.colegiosveterinarios.netcan.com:7051 CORE_P
EER_LOCALMSPID="ColegiosVeterinariosMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypt
o/peerOrganizations/colegiosveterinarios.netcan.com/peers/cvpaisvasco.colegiosveterinarios.netcan.com/tls/ca.crt peer channel
join -b netcanchannel.block
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/colegiosveterinarios.netca
n.com/users/Admin@colegiosveterinarios.netcan.com/msp/ CORE_PEER_ADDRESS=cvlarioja.colegiosveterinarios.netcan.com:7051 CORE_P
EER_LOCALMSPID="ColegiosVeterinariosMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
/peerOrganizations/colegiosveterinarios.netcan.com/peers/cvlarioja.colegiosveterinarios.netcan.com/tls/ca.crt peer channel join
-b netcanchannel.block

echo ""
echo "*****"
echo "Declarando los pares de anclaje"
echo "*****"

peer channel update -o orderer.netcan.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/FederacionesMSPanchors.tx --tls --
cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/m
sp/tlscacerts/tlsca.netcan.com-cert.pem
CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/colegiosveterinarios.netca
n.com/users/Admin@colegiosveterinarios.netcan.com/msp CORE_PEER_ADDRESS=cvmadrid.colegiosveterinarios.netcan.com:7051 CORE_P
EER_LOCALMSPID="ColegiosVeterinariosMSP" CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/p
eerOrganizations/colegiosveterinarios.netcan.com/peers/cvmadrid.colegiosveterinarios.netcan.com/tls/ca.crt peer channel update
-o orderer.netcan.com:7050 -c $CHANNEL_NAME -f ./channel-artifacts/ColegiosVeterinariosMSPanchors.tx --tls --
cafile /opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/m
sp/tlscacerts/tlsca.netcan.com-cert.pem

```

- chaincode_script.sh

Este script también se ejecuta dentro del docker del CLI e instala e instancia los chaincodes desarrollados

```
echo "Instalando e instanciando el chaincode $CHAINCODE"
echo "*****"
export CC_NOMBRE=$CHAINCODE
export CC_FILE=github.com/chaincode/netcan/$CHAINCODE/cc

peer chaincode install -n $CC_NOMBRE -v $CC_VERSION -p $CC_FILE
peer chaincode instantiate -n $CC_NOMBRE -v $CC_VERSION -c '{"Args":["init"]}' -o $ORDENADOR_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME -P "OR ('FederacionesMSP.peer', 'ColegiosVeterinariosMSP.peer')"
done
```

- json_script

Este script copia los archivos de carga iniciales a los dockers de los chaincodes para realizar las cargas iniciales de datos a la blockchain

```

#!/bin/bash

clear

echo "*****"
echo ""
echo "          /   /   / / - / / /   / / / / / /   / - / /   / / /   / /"
echo "         / / /   /   /   /   /   /   /   /   /   /   /   /   /   /   /"
echo "         /   /   /   /   /   /   /   /   /   /   /   /   /   /   /   /"
echo "         /   /   /   /   /   /   /   /   /   /   /   /   /   /   /   /"
echo "         /   /   /   /   /   /   /   /   /   /   /   /   /   /   /   /"
echo "         /   /   /   /   /   /   /   /   /   /   /   /   /   /   /   /"
echo "*****"
echo ""

echo "*****"
echo "Copiando los archivos json de carga inicial de datos"
echo "*****"

for DOCKER in $(docker ps --filter "since=cli" -q); do

    echo ""
    echo "*****"
    echo "Copiando los archivos json de carga inicial de datos al docker $DOCKER"
    echo "*****"

    docker cp /home/hyperledger/work/src/fabric-samples/TFM/json $DOCKER:./json
done

```

- `carga_script.sh`

Este script se ejecuta dentro del docker del CLI y realiza las cargas iniciales de datos a la blockchain

```
#!/bin/bash

export CHANNEL_NAME=netcanchannel
export CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

echo "*****"
echo "Carga inicial de datos"
echo "*****"

echo "*****"
echo "Cargando datos de PERFILES DE PERSONAS"
echo "*****"

peer chaincode invoke -n perfiles -
c '{"function":"cargarDatosIniciales","Args":["./json/perfiles.json", "{\"IDPersona\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -c $CHANNEL_NAME
```



```
peer chaincode invoke -n perfiles -  
c '{"function":"asignarEstado_OnlyAdmin","Args":["PERFILES_PERSONAS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":48}\", "{\"IDPersona\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n perfiles -  
c '{"function":"cancelarPerfilPersona","Args":["{\\"IDPersona\":0,\"CODPerfil\":\"ADMINISTRADOR\"}", "{$IDPersona}:0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de GRUPOS"  
echo "*****"  
  
peer chaincode invoke -n razas -c '{"function":"cargarDatosIniciales_Grupos","Args":[("./json/grupos.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n razas -  
c '{"function":"asignarEstado","Args":["GRUPOS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":12}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de RAZAS"  
echo "*****"  
  
peer chaincode invoke -n razas -c '{"function":"cargarDatosIniciales","Args":[("./json/razas.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n razas -  
c '{"function":"asignarEstado","Args":["RAZAS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":346}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de PERSONAS"  
echo "*****"  
  
peer chaincode invoke -n personas -c '{"function":"cargarDatosIniciales","Args":[("./json/personas_001.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s  
peer chaincode invoke -n personas -  
c '{"function":"asignarEstado","Args":["PERSONAS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":1000}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de AFIJOS"  
echo "*****"  
  
peer chaincode invoke -n afijos -c '{"function":"cargarDatosIniciales","Args":[("./json/afijos.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n afijos -  
c '{"function":"asignarEstado","Args":["AFIJOS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":50}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de PROPIETARIOS DE AFIJOS"  
echo "*****"  
  
peer chaincode invoke -n afijos -  
c '{"function":"cargarDatosIniciales_Propietarios","Args":[("./json/afijos_propietarios.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n afijos -  
c '{"function":"asignarEstado","Args":["AFIJOS_PROPIETARIOS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":50}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
  
echo "*****"  
echo "Cargando datos de PERROS"  
echo "*****"  
  
peer chaincode invoke -n perros -c '{"function":"cargarDatosIniciales","Args":[("./json/perros_001.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s  
peer chaincode invoke -n perros -c '{"function":"cargarDatosIniciales","Args":[("./json/perros_002.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s  
peer chaincode invoke -n perros -c '{"function":"cargarDatosIniciales","Args":[("./json/perros_003.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s  
peer chaincode invoke -n perros -c '{"function":"cargarDatosIniciales","Args":[("./json/perros_004.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s  
peer chaincode invoke -n perros -c '{"function":"cargarDatosIniciales","Args":[("./json/perros_005.json")]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
sleep 7s
```



```
peer chaincode invoke -n perros -
c '{"function":"asignarEstado","Args":["PERROS", "{\"docType\":\"CONTADOR\", \"IDMaximo\":5759}\"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME

echo "*****"
echo "Cargando datos de PROPIETARIOS DE PERROS"
echo "*****"

peer chaincode invoke -n perros -
c '{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_001.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n perros -
c '{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_002.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n perros -
c '{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_003.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n perros -
c '{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_004.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n perros -
c '{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_005.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n perros -
c '{"function":"asignarEstado","Args":["PERROS_PROPIETARIOS", "{\"docType\":\"CONTADOR\", \"IDMaximo\":5759}\"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME

echo "*****"
echo "Cargando datos de VETERINARIOS"
echo "*****"

peer chaincode invoke -n veterinarios -c '{"function":"cargarDatosIniciales","Args":["./json/veterinarios.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n veterinarios -
c '{"function":"asignarEstado","Args":["VETERINARIOS_PERSONAS", "{\"docType\":\"CONTADOR\", \"IDMaximo\":21}\"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME

echo "*****"
echo "Cargando datos de MICROCHIPS"
echo "*****"

peer chaincode invoke -n microchips -c '{"function":"cargarDatosIniciales","Args":["./json/microchips_perros_001.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n microchips -c '{"function":"cargarDatosIniciales","Args":["./json/microchips_perros_002.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n microchips -c '{"function":"cargarDatosIniciales","Args":["./json/microchips_perros_003.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n microchips -c '{"function":"cargarDatosIniciales","Args":["./json/microchips_perros_004.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n microchips -c '{"function":"cargarDatosIniciales","Args":["./json/microchips_perros_005.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n microchips -
c '{"function":"asignarEstado","Args":["MICROCHIPS_PERROS", "{\"docType\":\"CONTADOR\", \"IDMaximo\":5759}\"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME

echo "*****"
echo "Cargando datos de VACUNAS"
echo "*****"

peer chaincode invoke -n vacunas -
c '{"function":"cargarDatosIniciales_VacunasProteccion","Args":["./json/vacunas_proteccion_001.json"]}' -o $ORDENER_URL --tls --
cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s

echo "*****"
echo "Cargando datos de VACUNACIONES"
echo "*****"
```

```

peer chaincode invoke -n vacunas -
c '{"function":"asignarEstado","Args":["VACUNAS_PROTECCION", "{\"docType\":\"CONTADOR\",\"IDMaximo\":11}"]}' -o $ORDENER_URL --
-tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n vacunas -c '{"function":"cargarDatosIniciales","Args":["./json/vacunas_perros_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n vacunas -
c '{"function":"asignarEstado","Args":["VACUNAS_PERROS", "{\"docType\":\"CONTADOR\",\"IDMaximo\":89}"]}' -o $ORDENER_URL --
-tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n vacunas -
c '{"function":"cargarDatosIniciales_VacunasProteccion","Args":["./json/vacunas_perros_proteccion_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s
peer chaincode invoke -n vacunas -
c '{"function":"asignarEstado","Args":["VACUNAS_PERROS_PROTECCION", "{\"docType\":\"CONTADOR\",\"IDMaximo\":512}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

echo "*****"
echo "DATOS INICIALES CARGADOS"
echo "*****"

```

- stop netcan script

Por ultimo este script para la red, borra los dockers levantados y borra los chaincodes instanciados



CONFIGURACIÓN DE LA RED

Para realizar la configuración de la red es necesario definir los archivos mencionados en la estructura del proyecto situados bajo la carpeta TFM (configtx.yaml, crypto-config.yaml, docker-compose-cli.yaml y docker-compose-couch.yaml) así como los situados bajo la carpeta base (docker-compose-base.yaml y peer-base.yaml).

- configtx.yaml

En este archivo se realiza la configuración de las organizaciones y del canal y es el que servirá para la generación de los artefactos del proyecto.

Las secciones más importantes son:

- Organizations

Donde se definen las organizaciones con sus anchor peers, los directorios de configuración del MSP (Membership Service Provider) y las políticas de acceso para lectura, escritura y administración

Organizations:

```
# SampleOrg defines an MSP using the sampleconfig. It should never be used
# in production but may be used as a template for other definitions
- &OrdererOrg
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: OrdererOrg

    # ID to load the MSP definition as
    ID: OrdererMSP

    # MSPDir is the filesystem path which contains the MSP configuration
    MSPDir: crypto-config/ordererOrganizations/netcan.com/msp

    # Policies defines the set of policies at this level of the config tree
    # For organization policies, their canonical path is usually
    # /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Writers:
            Type: Signature
            Rule: "OR('OrdererMSP.member')"
        Admins:
            Type: Signature
            Rule: "OR('OrdererMSP.admin')"

- &Federaciones
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: FederacionesMSP

    # ID to load the MSP definition as
    ID: FederacionesMSP

    MSPDir: crypto-config/peerOrganizations/federaciones.netcan.com/msp

    # Policies defines the set of policies at this level of the config tree
    # For organization policies, their canonical path is usually
    # /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
        Readers:
            Type: Signature
```



```
        Rule: "OR('FederacionesMSP.admin', 'FederacionesMSP.peer', 'FederacionesMSP.client')"
Writers:
    Type: Signature
    Rule: "OR('FederacionesMSP.admin', 'FederacionesMSP.client')"
Admins:
    Type: Signature
    Rule: "OR('FederacionesMSP.admin')"

# leave this flag set to true.
AnchorPeers:
    # AnchorPeers defines the location of peers which can be used
    # for cross org gossip communication. Note, this value is only
    # encoded in the genesis block in the Application section context
    - Host: fci.federaciones.netcan.com
    Port: 7051

- &ColegiosVeterinarios
    # DefaultOrg defines the organization which is used in the sampleconfig
    # of the fabric.git development environment
    Name: ColegiosVeterinariosMSP

    # ID to load the MSP definition as
    ID: ColegiosVeterinariosMSP

    MSPDir: crypto-config/peerOrganizations/colegiosveterinarios.netcan.com/msp

    # Policies defines the set of policies at this level of the config tree
    # For organization policies, their canonical path is usually
    # /Channel/<Application|Orderer>/<OrgName>/<PolicyName>
    Policies:
        Readers:
            Type: Signature
            Rule: "OR('ColegiosVeterinariosMSP.admin', 'ColegiosVeterinariosMSP.peer', 'ColegiosVeterinariosMSP.client')"
        Writers:
            Type: Signature
            Rule: "OR('ColegiosVeterinariosMSP.admin', 'ColegiosVeterinariosMSP.client')"
        Admins:
            Type: Signature
            Rule: "OR('ColegiosVeterinariosMSP.admin')"

    AnchorPeers:
        # AnchorPeers defines the location of peers which can be used
        # for cross org gossip communication. Note, this value is only
        # encoded in the genesis block in the Application section context
        - Host: cvmadrid.colegiosveterinarios.netcan.com
        Port: 7051
```

- Orderer

Define el orderer y las políticas de acceso para lectura, escritura y administración

```
Orderer: &OrdererDefaults

    # Orderer Type: The orderer implementation to start
    # Available types are "solo" and "kafka"
    OrdererType: solo

    Addresses:
        - orderer.netcan.com:7050

    # Batch Timeout: The amount of time to wait before creating a batch
    BatchTimeout: 2s

    # Batch Size: Controls the number of messages batched into a block
    BatchSize:

        # Max Message Count: The maximum number of messages to permit in a batch
        MaxMessageCount: 100

        # Absolute Max Bytes: The absolute maximum number of bytes allowed for
        # the serialized messages in a batch.
        AbsoluteMaxBytes: 99 MB

        # Preferred Max Bytes: The preferred maximum number of bytes allowed for
        # the serialized messages in a batch. A message larger than the preferred
```



```
# max bytes will result in a batch larger than preferred max bytes.  
PreferredMaxBytes: 512 KB  
  
Kafka:  
# Brokers: A list of Kafka brokers to which the orderer connects  
# NOTE: Use IP:port notation  
Brokers:  
- 127.0.0.1:9092  
  
# Organizations is the list of orgs which are defined as participants on  
# the orderer side of the network  
Organizations:  
  
# Policies defines the set of policies at this level of the config tree  
# For Orderer policies, their canonical path is  
# /Channel/Orderer/<PolicyName>  
Policies:  
Readers:  
Type: ImplicitMeta  
Rule: "ANY Readers"  
Writers:  
Type: ImplicitMeta  
Rule: "ANY Writers"  
Admins:  
Type: ImplicitMeta  
Rule: "MAJORITY Admins"  
# BlockValidation specifies what signatures must be included in the block  
# from the orderer for the peer to validate it.  
BlockValidation:  
Type: ImplicitMeta  
Rule: "ANY Writers"
```

- Channel & profiles

Define el canal, las organizaciones, el consorcio,...

```
Channel: &ChannelDefaults  
# Policies defines the set of policies at this level of the config tree  
# For Channel policies, their canonical path is  
# /Channel/<PolicyName>  
Policies:  
# Who may invoke the 'Deliver' API  
Readers:  
Type: ImplicitMeta  
Rule: "ANY Readers"  
# Who may invoke the 'Broadcast' API  
Writers:  
Type: ImplicitMeta  
Rule: "ANY Writers"  
# By default, who may modify elements at this config level  
Admins:  
Type: ImplicitMeta  
Rule: "MAJORITY Admins"  
  
# Capabilities describes the channel level capabilities, see the  
# dedicated Capabilities section elsewhere in this file for a full  
# description  
Capabilities:  
<>: *ChannelCapabilities  
  
#####  
#  
# Profile  
#  
# - Different configuration profiles may be encoded here to be specified  
# as parameters to the configtxgen tool  
#  
#####  
Profiles:  
  
NetCanOrdererGenesis:  
<>: *ChannelDefaults  
Orderer:  
<>: *OrdererDefaults  
Organizations:  
- *OrdererOrg
```



```
Capabilities:  
  <<: *OrdererCapabilities  
Consortiums:  
  NetCanConsortium :  
    Organizations:  
      - *Federaciones  
      - *ColegiosVeterinarios  
NetCanChannel:  
  Consortium: NetCanConsortium  
  Application:  
    <<: *ApplicationDefaults  
    Organizations:  
      - *Federaciones  
      - *ColegiosVeterinarios  
    Capabilities:  
      <<: *ApplicationCapabilities  
  
NetCanDevModeKafka:  
  <<: *ChannelDefaults  
  Capabilities:  
    <<: *ChannelCapabilities  
  Orderer:  
    <<: *OrdererDefaults  
    OrdererType: kafka  
    Kafka:  
      Brokers:  
        - kafka.netcan.com:9092  
  
      Organizations:  
        - *OrdererOrg  
      Capabilities:  
        <<: *OrdererCapabilities  
  Application:  
    <<: *ApplicationDefaults  
    Organizations:  
      - <<: *OrdererOrg  
  Consortiums:  
  NetCanConsortium:  
    Organizations:  
      - *Federaciones  
      - *ColegiosVeterinarios
```

- `crypto-config.yaml`

Define la arquitectura de la red con el orderer, las organizaciones y los peers para la generación del material criptográfico por medio de la herramienta de hyperledger fabric cryptogen

```
# Copyright IBM Corp. All Rights Reserved.  
#  
# SPDX-License-Identifier: Apache-2.0  
#  
# -----  
# "OrdererOrgs" - Definition of organizations managing orderer nodes  
# -----  
OrdererOrgs:  
  
  - Name: Orderer  
    Domain: netcan.com  
  
    Specs:  
      - Hostname: orderer  
# -----  
# "PeerOrgs" - Definition of organizations managing peer nodes  
# -----  
PeerOrgs:  
  
  - Name: Federaciones  
    Domain: federaciones.netcan.com  
    EnableNodeOUs: true  
  
    Specs:  
      - Hostname: fci
```



```
- Hostname: rsce
- Hostname: tkc
- Hostname: acw

Users:
Count: 1

- Name: ColegiosVeterinarios
Domain: colegiosveterinarios.netcan.com
EnableNodeOUs: true
Specs:
- Hostname: cvmadrid
- Hostname: cvandalucia
- Hostname: cvaragon
- Hostname: cvasturias
- Hostname: cvillesbalears
- Hostname: cvcanarias
- Hostname: cvcantabria
- Hostname: cvcastillayleon
- Hostname: cvcastillalamancha
- Hostname: cvcataluna
- Hostname: cvcomunitatvalenciana
- Hostname: cvextremadura
- Hostname: cvgalicia
- Hostname: cvmurcia
- Hostname: cvnavarra
- Hostname: cvpaisvasco
- Hostname: cvlarioja
- Hostname: cvceuta
- Hostname: cvmelilla

Users:
Count: 1
```

- docker-compose-cli.yaml

Es el archivo de configuración principal de Hyperledger Fabric. En él se define la estructura básica del proyecto.

En primer lugar se definen las CA's (certíficate authorities).

En este caso irán sin TLS (será lo único sin TLS del proyecto securizado ya que en conversaciones con el tutor nos ha recomendado no securizarlas ya que no es necesario) y se define también el puerto de las mismas y el comando para inicializarlas en el que hay que introducir la ca.certfile y la ca.keyfile correspondientes que se encuentran en la carpeta de la organización en peerOrganizations en la carpeta crypto-config.

A continuación se definirán el orderer y los peers, que extenderán de los archivos situados en la carpeta base.

Por último se define el CLI, que irá securizado con TLS por lo que se le pasarán los certificados correspondientes en las variables CORE_PEER_TLS_CERT_FILE, CORE_PEER_TLS_KEY_FILE y CORE_PEER_TLS_ROOTCERT_FILE



```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

volumes:
  orderer.netcan.com:
  fci.federaciones.netcan.com:
  rsce.federaciones.netcan.com:
  tkc.federaciones.netcan.com:
  acw.federaciones.netcan.com:
  cvandalucia.colegiosveterinarios.netcan.com:
  cvaragon.colegiosveterinarios.netcan.com:
  cvasturias.colegiosveterinarios.netcan.com:
  cvillesbalears.colegiosveterinarios.netcan.com:
  cvcanarias.colegiosveterinarios.netcan.com:
  cvcantabria.colegiosveterinarios.netcan.com:
  cvcastillaleon.colegiosveterinarios.netcan.com:
  cvcataluna.colegiosveterinarios.netcan.com:
  cvcomunitatvalenciana.colegiosveterinarios.netcan.com:
  cvextremadura.colegiosveterinarios.netcan.com:
  cvgalicia.colegiosveterinarios.netcan.com:
  cvmadrid.colegiosveterinarios.netcan.com:
  cvmurcia.colegiosveterinarios.netcan.com:
  cvnavarra.colegiosveterinarios.netcan.com:
  cvpaisvasco.colegiosveterinarios.netcan.com:
  cvlarioja.colegiosveterinarios.netcan.com:

networks:
  netcan:

services:

  ca.federaciones:
    image: hyperledger/fabric-ca:1.4.0
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-federaciones
      - FABRIC_CA_SERVER_TLS_ENABLED=false
      - FABRIC_CA_SERVER_PORT=7054
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.federaciones.netcan.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/ee7e9f81bf8ddf11193119ffb4eb756a47d0aa326b921ff020179c30408826d_sk -b admin:adminpw -d --cfg.identities.allowremove'
    volumes:
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
      - /home/hyperledger/work/src/fabric-samples/TFM/crypto-
config/peerOrganizations/federaciones.netcan.com/ca:/etc/hyperledger/fabric-ca-server-config
    container_name: ca_federaciones
    networks:
      - netcan

  ca.colegiosveterinarios:
    image: hyperledger/fabric-ca:1.4.0
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-colegiosveterinarios
      - FABRIC_CA_SERVER_TLS_ENABLED=false
      - FABRIC_CA_SERVER_PORT=7054
    ports:
      - "8054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.colegiosveterinarios.netcan.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/c605e45a562d0c01566a2e3bc452e653d7b92af1c04fdb0a79263b8df4dc0a7a_sk -b admin-cv:admin-cvpw -d --cfg.identities.allowremove'
    volumes:
      - ./crypto-config/peerOrganizations/colegiosveterinarios.netcan.com/ca:/etc/hyperledger/fabric-ca-server-config
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    container_name: ca_colegiosveterinarios
    networks:
      - netcan

  orderer.netcan.com:
    extends:
      file: base/docker-compose-base.yaml
      service: orderer.netcan.com
```



```
container_name: orderer.netcan.com
networks:
  - netcan

fci.federaciones.netcan.com:
  container_name: fci.federaciones.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: fci.federaciones.netcan.com
  networks:
    - netcan

rsce.federaciones.netcan.com:
  container_name: rsce.federaciones.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: rsce.federaciones.netcan.com
  networks:
    - netcan

tkc.federaciones.netcan.com:
  container_name: tkc.federaciones.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: tkc.federaciones.netcan.com
  networks:
    - netcan

acw.federaciones.netcan.com:
  container_name: acw.federaciones.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: acw.federaciones.netcan.com
  networks:
    - netcan

cvandalucia.colegiosveterinarios.netcan.com:
  container_name: cvandalucia.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvandalucia.colegiosveterinarios.netcan.com
  networks:
    - netcan

cvaragon.colegiosveterinarios.netcan.com:
  container_name: cvaragon.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvaragon.colegiosveterinarios.netcan.com
  networks:
    - netcan

cvasturias.colegiosveterinarios.netcan.com:
  container_name: cvasturias.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvasturias.colegiosveterinarios.netcan.com
  networks:
    - netcan

cvillesbalears.colegiosveterinarios.netcan.com:
  container_name: cvillesbalears.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvillesbalears.colegiosveterinarios.netcan.com
  networks:
    - netcan

cvcnarias.colegiosveterinarios.netcan.com:
  container_name: cvcnarias.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvcnarias.colegiosveterinarios.netcan.com
  networks:
    - netcan

cvcantabria.colegiosveterinarios.netcan.com:
  container_name: cvcantabria.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
```



```
service: cvcantabria.colegiosveterinarios.netcan.com
networks:
- netcan

cvcastillayleon.colegiosveterinarios.netcan.com:
  container_name: cvcastillayleon.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvcastillayleon.colegiosveterinarios.netcan.com
  networks:
- netcan

cvcastillalamancha.colegiosveterinarios.netcan.com:
  container_name: cvcastillalamancha.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvcastillalamancha.colegiosveterinarios.netcan.com
  networks:
- netcan

cvcataluna.colegiosveterinarios.netcan.com:
  container_name: cvcataluna.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvcataluna.colegiosveterinarios.netcan.com
  networks:
- netcan

cvcomunitatvalenciana.colegiosveterinarios.netcan.com:
  container_name: cvcomunitatvalenciana.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvcomunitatvalenciana.colegiosveterinarios.netcan.com
  networks:
- netcan

cvextremadura.colegiosveterinarios.netcan.com:
  container_name: cvextremadura.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvextremadura.colegiosveterinarios.netcan.com
  networks:
- netcan

cvgalicia.colegiosveterinarios.netcan.com:
  container_name: cvgalicia.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvgalicia.colegiosveterinarios.netcan.com
  networks:
- netcan

cvmadrid.colegiosveterinarios.netcan.com:
  container_name: cvmadrid.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvmadrid.colegiosveterinarios.netcan.com
  networks:
- netcan

cvmurcia.colegiosveterinarios.netcan.com:
  container_name: cvmurcia.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvmurcia.colegiosveterinarios.netcan.com
  networks:
- netcan

cvnavarra.colegiosveterinarios.netcan.com:
  container_name: cvnavarra.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvnavarra.colegiosveterinarios.netcan.com
  networks:
- netcan

cvpaisvasco.colegiosveterinarios.netcan.com:
  container_name: cvpaisvasco.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
```



```
service: cvpaisvasco.colegiosveterinarios.netcan.com
networks:
- netcan

cvlarioja.colegiosveterinarios.netcan.com:
  container_name: cvlarioja.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: cvlarioja.colegiosveterinarios.netcan.com
  networks:
- netcan

cli:
  container_name: cli
  image: hyperledger/fabric-tools:1.4.0
  tty: true
  stdin_open: true
  environment:
    - GOPATH=/opt/gopath
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - FABRIC_LOGGING_SPEC=INFO
    - CORE_PEER_ID=cli
    - CORE_PEER_ADDRESS=fci.federaciones.netcan.com:7051
    - CORE_PEER_LOCALMSPID=FederacionesMSP
    - CORE_PEER_TLS_ENABLED=true
    -
    CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/fci.federaciones.netcan.com/tls/server.crt
    -
    CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/fci.federaciones.netcan.com/tls/server.key
    -
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/fci.federaciones.netcan.com/tls/ca.crt
    -
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/users/Admin@federaciones.netcan.com/msp
      working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
      command: /bin/bash
  volumes:
    - /var/run/:/host/var/run/
    - /home/hyperledger/work/src/fabric-samples/TFM/chaincode/:/opt/gopath/src/github.com/chaincode/
    - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    - ./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/
    - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts
depends_on:
- orderer.netcan.com
- fci.federaciones.netcan.com
- rsce.federaciones.netcan.com
- tkc.federaciones.netcan.com
- acw.federaciones.netcan.com
- cvandalucia.colegiosveterinarios.netcan.com
- cvaragon.colegiosveterinarios.netcan.com
- cvasturias.colegiosveterinarios.netcan.com
- cvillesbalears.colegiosveterinarios.netcan.com
- cvcanarias.colegiosveterinarios.netcan.com
- cvcantabria.colegiosveterinarios.netcan.com
- cvcastillayleon.colegiosveterinarios.netcan.com
- cvcastillalamancha.colegiosveterinarios.netcan.com
- cvcataluna.colegiosveterinarios.netcan.com
- cvcomunitatvalenciana.colegiosveterinarios.netcan.com
- cvextremadura.colegiosveterinarios.netcan.com
- cvgalicia.colegiosveterinarios.netcan.com
- cvmadrid.colegiosveterinarios.netcan.com
- cvmurcia.colegiosveterinarios.netcan.com
- cvnavarra.colegiosveterinarios.netcan.com
- cvpaisvasco.colegiosveterinarios.netcan.com
- cvlarioja.colegiosveterinarios.netcan.com
networks:
- netcan
```



- docker-compose-couch.yaml

Este archivo configura la red para el uso de CouchDB en vez de levelDB, lo que permite realizar índices y redonda en una mayor facilidad de uso y la posibilidad de realización de consultas más complejas.

En este caso no se define usuario y contraseña por facilidad de uso al ser seis componentes en el grupo de desarrollo, aunque en producción deberá definirse un usuario y contraseña para la couchDB.

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

networks:
  netcan:

services:
  couchdb0:
    container_name: couchdb0
    image: hyperledger/fabric-couchdb:0.4.15
    # Populate the COUCHDB_USER and COUCHDB_PASSWORD to set an admin user and password
    # for CouchDB. This will prevent CouchDB from operating in an "Admin Party" mode.
    environment:
      - COUCHDB_USER=netcan
      - COUCHDB_PASSWORD=cannet
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    # Comment/Uncomment the port mapping if you want to hide/expose the CouchDB service,
    # for netcan map it to utilize Fauxton User Interface in dev environments.
    ports:
      - "5984:5984"
    networks:
      - netcan

  fci.federaciones.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb0

  couchdb1:
    container_name: couchdb1
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "6984:5984"
    networks:
      - netcan

  rsce.federaciones.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb1

  couchdb2:
  .
  .
```



- docker-compose-base.yaml

En este archivo se definen el orderer y los peers. En particular se establece que los peers extienden a su vez del archivo peer-base.yaml y el uso de TLS en el orderer pasando los certificados correspondientes en las variables ORDERER_GENERAL_TLS_PRIVATEKEY, ORDERER_GENERAL_TLS_CERTIFICATE y ORDERER_GENERAL_TLS_ROOTCAS

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

services:

  orderer.netcan.com:
    container_name: orderer.netcan.com
    image: hyperledger/fabric-orderer:1.4.0
    environment:
      - ORDERER_GENERAL_LOGLEVEL=INFO
      - ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
      - ORDERER_GENERAL_GENESISMETHOD=file
      - ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
      - ORDERER_GENERAL_LOCALMSPID=OrdererMSP
      - ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/orderer/msp
    # Enabled TLS
    - ORDERER_GENERAL_TLS_ENABLED=true
    - ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
    - ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
    - ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
    - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=tfm_netcan
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric
    command: orderer
    volumes:
      - ./channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block
      - ./crypto-config/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp:/var/hyperledger/orderer/msp
      - ./crypto-config/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/tls:/var/hyperledger/orderer/tls
      - orderer.netcan.com:/var/hyperledger/production/orderer
    ports:
      - 7050:7050

  fci.federaciones.netcan.com:
    container_name: fci.federaciones.netcan.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=fci.federaciones.netcan.com
      - CORE_PEER_ADDRESS=fci.federaciones.netcan.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=rsce.federaciones.netcan.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=fci.federaciones.netcan.com:7051
      - CORE_PEER_LOCALMSPID=FederacionesMSP
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/fci.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
      - ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/fci.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
      - fci.federaciones.netcan.com:/var/hyperledger/production
    ports:
      - 7051:7051
      - 7053:7053

  rsce.federaciones.netcan.com:
    container_name: rsce.federaciones.netcan.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=rsce.federaciones.netcan.com
      - CORE_PEER_ADDRESS=rsce.federaciones.netcan.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=rsce.federaciones.netcan.com:7051
```



```
- CORE_PEER_GOSSIP_BOOTSTRAP=fci.federaciones.netcan.com:7051
- CORE_PEER_LOCALMSPID=FederacionesMSP
volumes:
  - /var/run/:/host/var/run/
  - ..../crypto-
config/peerOrganizations/federaciones.netcan.com/peers/rsce.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ..../crypto-
config/peerOrganizations/federaciones.netcan.com/peers/rsce.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
  - rsce.federaciones.netcan.com:/var/hyperledger/production
ports:
  - 8051:7051
  - 8053:7053
.
.
```

- **peer-base.yaml**

Establece la configuración común a todos los peers, incluido el uso de TLS y sus correspondientes certificados

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

services:
  peer-base:
    image: hyperledger/fabric-peer:1.4.0
    environment:
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=tfm_netcan
      - FABRIC_LOGGING_SPEC=INFO
      - CORE_PEER_TLS_ENABLED=true
      - CORE_PEER_GOSSIP_USELEADERELECTION=true
      - CORE_PEER_GOSSIP_ORGLEADER=false
      - CORE_PEER_PROFILE_ENABLED=true
      - CORE_PEER_ENDORSER_ENABLED=true
      - CORE_PEER_TLS_CERT_FILE=/etc/hyperledger/fabric/tls/server.crt
      - CORE_PEER_TLS_KEY_FILE=/etc/hyperledger/fabric/tls/server.key
      - CORE_PEER_TLS_ROOTCERT_FILE=/etc/hyperledger/fabric/tls/ca.crt
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: peer node start
```



Puesta en marcha del proyecto

RED INICIAL SIN TLS

Los archivos de configuración pueden encontrarse en el repositorio Github mencionado anteriormente:

https://github.com/DFLBB/TFM_archs

En la carpeta Arquitectura_bolckchain/1_server_without_TLS.

Se trata de una copia completa del proyecto.

Los scripts utilizados son los que se encuentran en la carpeta scripts2 del repositorio github.

Lo más relevante de esta versión es que no utiliza TLS, tal y como puede verse en los archivos de configuración docker-compose-cli.yaml, docker-compose-base.yaml y peer-base.yaml:

```
cli:
  container_name: cli
  image: hyperledger/fabric-tools:1.4.0
  tty: true
  stdin_open: true
  environment:
    - GOPATH=/opt/gopath
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - FABRIC_LOGGING_SPEC=INFO
    - CORE_PEER_ID=cli
    - CORE_PEER_ADDRESS=fci.federaciones.netcan.com:7051
    - CORE_PEER_LOCALMSPID=FederacionesMSP
    - CORE_PEER_TLS_ENABLED=false
    -
  CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/users/Admin@federaciones.netcan.com/msp
  working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
```

Una vez instalados los prerequisitos y preparado el servidor para el uso de Hyperledger Fabric y una vez preparada la carpeta del proyecto en la que tendremos los archivos de configuración, la carpeta base con sus archivos de configuración, la carpeta cannel-artifacts vacía (únicamente con un archivo .gitkeep vacío) y los scripts de inicio y parada del proyecto comenzamos con la instalación de la red.

Desde la carpeta del proyecto situada en /home/hyperledger/work/src/fabric-samples/TFM lanzaremos los siguientes comandos:



```
export PATH=/home/hyperledger/work/src/fabric-samples/bin:$PATH
# Generar material criptográfico /// NO HACER SI SE REINICIA Y NO SE QUIERE BORRAR
../bin/cryptogen generate --config=./crypto-config.yaml
export FABRIC_CFG_PATH=$PWD
# Generar EL bloque génesis
../bin/configtxgen -profile NetCanOrdererGenesis -outputBlock ./channel-artifacts/genesis.block

# Generar la configuración del canal
export CHANNEL_NAME=netcanchannel &&../bin/configtxgen -profile NetCanChannel -
outputCreateChannelTx ./channel-artifacts/channel.tx -channelID $CHANNEL_NAME

# Definir los pares de anclaje
../bin/configtxgen -profile NetCanChannel -outputAnchorPeersUpdate ./channel-
artifacts/FederacionesMSPanchors.tx -channelID $CHANNEL_NAME -asOrg FederacionesMSP

../bin/configtxgen -profile NetCanChannel -outputAnchorPeersUpdate ./channel-
artifacts/ColegiosVeterinariosMSPanchors.tx -channelID $CHANNEL_NAME -asOrg ColegiosVeterinariosMSP
```

En este punto se ha generado el material criptográfico en una carpeta llamada crypto-config y se han generado los artefactos en la carpeta channel-artifacts y podemos llamar al script netcan_script.sh que levantará y configurará la red y los chaincodes desarrollados utilizando el resto de scripts explicados anteriormente.

```
• MobaXterm 20.1 •
(SSH client, X-server and networking tools)

> SSH session to hyperledger@82.223.205.242
  • SSH compression : ✓
  • SSH-browser : ✓
  • X11-forwarding : ✓ (remote display is forwarded through SSH)
  • DISPLAY : ✓ (automatically set on remote server)

> For more info, ctrl+click on help or visit our website

Welcome to Ubuntu 16.04.7 LTS (GNU/Linux 4.4.0-187-generic x86_64)

 * Documentation: https://help.ubuntu.com
 * Management: https://landscape.canonical.com
 * Support: https://ubuntu.com/advantage

* Kubernetes 1.19 is out! Get it in one command with:

  sudo snap install microk8s --channel=1.19 --classic

  https://microk8s.io/ has docs and details.
Last login: Sat Sep 12 12:34:08 2020 from 95.122.217.108
hyperledger@localhost:~$ cd /home/hyperledger/work/src/fabric-samples/TFM/
hyperledger@localhost:~/work/src/fabric-samples/TFM$ ./netcan_script.sh
```



```
*****  
Bienvenido a  
*****  
  
*****  
Limpiando la instalación  
*****  
"docker stop" requires at least 1 argument.  
See 'docker stop --help'.  
  
Usage: docker stop [OPTIONS] CONTAINER [CONTAINER...]  
  
Stop one or more running containers  
"docker rm" requires at least 1 argument.  
See 'docker rm --help'.  
  
Usage: docker rm [OPTIONS] CONTAINER [CONTAINER...]  
  
Remove one or more containers  
Total reclaimed space: 0B  
Total reclaimed space: 0B  
Cloning into '/home/hyperledger/work/src/fabric-samples/TFM/TFM_archs'...
```

La red así levantada no lleva TLS, tiene implementada la CouchDB y tiene implementadas las CA's.

```
*****  
*****  
Estableciendo las variables de entorno del canal  
*****  
  
*****  
Creando el canal  
*****  
2020-09-12 12:37:21.892 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:37:22.251 UTC [cli.common] readBlock -> INFO 002 Received block: 0
```

Se crea el canal

```
*****  
Adhiriendo el peer FCI de Federaciones al canal  
Puede tardar un poco...  
*****  
2020-09-12 12:38:23.329 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:38:23.492 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel  
*****  
Adhiriendo el resto de peers de ambas organizaciones al canal  
*****  
2020-09-12 12:38:24.698 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:38:25.254 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel  
2020-09-12 12:38:25.454 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:38:25.577 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel  
2020-09-12 12:38:25.668 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:38:25.833 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel  
2020-09-12 12:38:25.948 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized  
2020-09-12 12:38:26.111 UTC [channelCmd] executeJoin -> INFO 002 Successfully submitted proposal to join channel  
2020-09-12 12:38:26.185 UTC [channelCmd] InitCmdFactory -> INFO 001 Endorser and orderer connections initialized
```

Se adhieren los peers al canal.

Se instalan e instancian los chaincodes

```
*****
Carga inicial de datos
*****
*****
Cargando datos de PERFILES DE PERSONAS
*****
2020-09-12 12:43:28.443 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"48"
2020-09-12 12:43:28.622 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\\"docType\\":\\"CONTADOR\\",\\"IDMaximo\\":48}"
2020-09-12 12:43:28.732 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\\"docType\\":\\"PERFILES_PERSONAS\\",\\"IDPerfilPersona\\":0,\\"IDPersona\\":0,\\"CODPerfil\\":\\"ADMINISTRADOR\\",\\"FechaAlta\\":\\"2020-09-12 12:40:11.817017251 +0000 UTC m=+0.113422036\\",\\"FechaBaja\\":\\"2020-09-12 12:43:28.710732619 +0000 UTC m=+197.007137410\\\"}"
*****
Cargando datos de GRUPOS
*****
2020-09-12 12:43:28.950 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"12"
2020-09-12 12:43:29.154 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\\"docType\\":\\"CONTADOR\\",\\"IDMaximo\\":12}"
*****
Cargando datos de RAZAS
*****
2020-09-12 12:43:29.706 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"346"
2020-09-12 12:43:29.816 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\\"docType\\":\\"CONTADOR\\",\\"IDMaximo\\":346}"
*****
Cargando datos de PERSONAS
*****
```

Y se realiza la carga inicial de datos a la blockchain

Al estar implementada la CouchDB en el puerto 5984 podemos conectarnos para ver si todas las cargas han ido correctamente y están los datos (a pesar de que al realizar las cargas la CouchDB responde con un “OK”), comprobando que efectivamente todos los datos se han cargado a la CouchDB.



The screenshot shows the MongoDB Compass interface with a sidebar on the left containing icons for Databases, Collections, and Tools. The main area displays a table titled 'Databases' with the following columns: Name, Size, # of Docs, and Actions. The table lists 13 databases:

Name	Size	# of Docs	Actions
_replicator	2.3 kB	1	⋮ ⚡ 🔒 ⌂
_users	2.3 kB	1	⋮ ⚡ 🔒 ⌂
netcanchannel_	18.4 kB	2	⋮ ⚡ 🔒 ⌂
netcanchannel_alpha	31.4 kB	106	⋮ ⚡ 🔒 ⌂
netcanchannel_ccc	6.0 kB	9	⋮ ⚡ 🔒 ⌂
netcanchannel_microchips	1.5 MB	4762	⋮ ⚡ 🔒 ⌂
netcanchannel_perfiles	16.3 kB	51	⋮ ⚡ 🔒 ⌂
netcanchannel_perros	3.1 MB	9524	⋮ ⚡ 🔒 ⌂
netcanchannel_personas	385.8 kB	1003	⋮ ⚡ 🔒 ⌂
netcanchannel razas	103.9 kB	352	⋮ ⚡ 🔒 ⌂
netcanchannel_solicitudes	0.5 kB	1	⋮ ⚡ 🔒 ⌂
netcanchannel_vacunas	37.5 kB	104	⋮ ⚡ 🔒 ⌂
netcanchannel_veterinarios	8.5 kB	24	⋮ ⚡ 🔒 ⌂

At the bottom right, it says 'Showing 1-13 of 13 databases'.

Pudiendo ver los datos cargados en cada tabla:

The screenshot shows the MongoDB Compass interface with a sidebar on the left containing icons for Databases, Collections, and Tools. The main area displays a table titled 'All Documents' for the 'netcanchannel_perros' database. The table has the following columns: _id, _rev, IDMaximo, docType, ~version, FechaAlta, FechaBaja, FechaDefuncion, FechaNacimiento, IdObjeto, IDPerro, IDPerroMadre, IDPerroPadre, IDRaza, IDSexo, and Nombre. The table lists numerous documents representing dogs, each with unique IDs and various attributes like name, birth date, and breed. At the bottom, it says 'Showing 16 columns' and 'Showing document 1 - 100'. There is also a 'Documents per page' dropdown set to 100.

Podemos comprobar que todos los dockers están corriendo correctamente mediante el comando:

```
docker ps -a
```

Obteniendo la siguiente salida en la que puede comprobarse que la red está corriendo correctamente:

```
hyperledger@localhost:~/work/src/fabric-samples/TFM$ docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              NAMES
847766e969dc        dev-fci.federaciones.netcan.com-veterinarios-1.0.0-   Up
130c44c62ff0205de3b79eea6e6c6fdb54e461931866f81fb4bf3ad7300101fd   "chaincode -peer.add..."   13 minutes ago
13 minutes           dev-fci.federaciones.netcan.com-veterinarios-1.0.0
8d4de91db75c        dev-fci.federaciones.netcan.com-vacunas-1.0.0-      Up
c0712aaa2c66f0eb4d3e68b9dca83fb740be117b8e8c9e54b50a94da72f3f366   "chaincode -peer.add..."   14 minutes ago
Up 14 minutes        dev-fci.federaciones.netcan.com-vacunas-1.0.0
969cceaa9c3a4        dev-fci.federaciones.netcan.com-solicitudes-1.0.0-    Up
de8c71563a9bf41548efc3efd696676c22d66525e1f312b37574d60c1b17aab0   "chaincode -peer.add..."   14 minutes ago
Up 14 minutes        dev-fci.federaciones.netcan.com-solicitudes-1.0.0
```



ab54d7839973		dev-fci.federaciones.netcan.com-razas-1.0.0-		
b5e08411bd92c32ee9f31c54d803da0da9dbc07c35b86bdd9dbc66c7d9379d6c	"chaincode -peer.add..."	15 minutes ago		
Up 15 minutes	dev-fci.federaciones.netcan.com-razas-1.0.0			
91ade708082e	dev-fci.federaciones.netcan.com-personas-1.0.0-			
ec9fd3c3016785411df3b0cfb73fb0c4a2ff8f249ada3ae8604d52579cf2ca92	"chaincode -peer.add..."	15 minutes ago	Up	
15 minutes	dev-fci.federaciones.netcan.com-personas-1.0.0			
8e8a58923aae	dev-fci.federaciones.netcan.com-perros-1.0.0-			
46b41e94180f7b7b9d3a25abd665310a9fe68ed5639e9ab474ddf30c60d278f3	"chaincode -peer.add..."	16 minutes ago		
Up 16 minutes	dev-fci.federaciones.netcan.com-perros-1.0.0			
3a2a8692d58f	dev-fci.federaciones.netcan.com-perfiles-1.0.0-			
ee62113a5af70130e6a0df3310ec7555afcaee7f3659eb7a4191efdf56bfa5d8	"chaincode -peer.add..."	17 minutes ago		
Up 17 minutes	dev-fci.federaciones.netcan.com-perfiles-1.0.0			
ffb55ddfcc92	dev-fci.federaciones.netcan.com-microchips-1.0.0-			
fb61a6a31883fa2a1b3280a342942d5462730c2b4264e4e39e8f7cec9785092c	"chaincode -peer.add..."	17 minutes ago		
Up 17 minutes	dev-fci.federaciones.netcan.com-microchips-1.0.0			
e4be7a159d3e	dev-fci.federaciones.netcan.com-afijos-1.0.0-			
33d55e901eadaa70caea32134c737a7fc4caa1ca66c9f8a937444c5383c8dbe	"chaincode -peer.add..."	18 minutes ago		
Up 18 minutes	dev-fci.federaciones.netcan.com-afijos-1.0.0			
49f928e0ac99	hyperledger/fabric-tools:1.4.0		"/bin/bash"	19
minutes ago	Up 19 minutes	cli		
eef9afbc3c31	hyperledger/fabric-peer:1.4.0		"peer node start"	20
minutes ago	Up 19 minutes	0.0.0.0:19051->7051/tcp,	0.0.0.0:19053->7053/tcp	
cvcastillalamancha.colegiosveterinarios.netcan.com				
695f8a37dd86	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:18051->7051/tcp,	0.0.0.0:18053->7053/tcp	
cvcastillayleon.colegiosveterinarios.netcan.com				
6c8455958aaaf	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:14051->7051/tcp,	0.0.0.0:14053->7053/tcp	
cvasturias.colegiosveterinarios.netcan.com				
aa271f0e3eaa	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:21051->7051/tcp,	0.0.0.0:21053->7053/tcp	
cvcomunitatvalenciana.colegiosveterinarios.netcan.com				
91468eeb882c	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:17051->7051/tcp,	0.0.0.0:17053->7053/tcp	
cvcantabria.colegiosveterinarios.netcan.com				
7c0675f67108	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:24051->7051/tcp,	0.0.0.0:24053->7053/tcp	
cvmurcia.colegiosveterinarios.netcan.com				
6254cb2d73bd	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:11051->7051/tcp,	0.0.0.0:11053->7053/tcp	
cvmadrid.colegiosveterinarios.netcan.com				
f9aeb8c5ea47	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:26051->7051/tcp,	0.0.0.0:26053->7053/tcp	
cvpaisvasco.colegiosveterinarios.netcan.com				
2a5892757a6f	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:20051->7051/tcp,	0.0.0.0:20053->7053/tcp	
cvcataluna.colegiosveterinarios.netcan.com				
ca243207d8f3	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:25051->7051/tcp,	0.0.0.0:25053->7053/tcp	
cnavarra.colegiosveterinarios.netcan.com				
4a028d76740d	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:27051->7051/tcp,	0.0.0.0:27053->7053/tcp	
cvlarioja.colegiosveterinarios.netcan.com				
3a24bd3fcfd56	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:16051->7051/tcp,	0.0.0.0:16053->7053/tcp	
cvcnarias.colegiosveterinarios.netcan.com				
2979b1ac8015	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 19 minutes	0.0.0.0:9051->7051/tcp,	0.0.0.0:9053->7053/tcp	tkc.federaciones.netcan.com
45b835a6641a	hyperledger/fabric-peer:1.4.0		"peer node start"	
20 minutes ago	Up 20 minutes	0.0.0.0:15051->7051/tcp,	0.0.0.0:15053->7053/tcp	
civillesbalears.colegiosveterinarios.netcan.com				
7de244f6f55f	hyperledger/fabric-peer:1.4.0		"peer node start"	20
minutes ago	Up 19 minutes	0.0.0.0:7051->7051/tcp,	0.0.0.0:7053->7053/tcp	fci.federaciones.netcan.com



5315084811fa	hyperledger/fabric-peer:1.4.0				"peer node start"
20 minutes ago	Up 19 minutes		0.0.0.0:22051->7051/tcp, 0.0.0.0:22053->7053/tcp		
cvextremadura.colegiosveterinarios.netcan.com					
286bb539e7e8	hyperledger/fabric-peer:1.4.0				"peer node start"
20 minutes ago	Up 19 minutes		0.0.0.0:13051->7051/tcp, 0.0.0.0:13053->7053/tcp		
cvaragon.colegiosveterinarios.netcan.com					
0aca3174afbe	hyperledger/fabric-peer:1.4.0				"peer node start"
20 minutes ago	Up 19 minutes	0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp	rsce.federaciones.netcan.com		
21697bb1fac3	hyperledger/fabric-peer:1.4.0				"peer node start"
20 minutes ago	Up 20 minutes	0.0.0.0:10051->7051/tcp, 0.0.0.0:10053->7053/tcp	acw.federaciones.netcan.com		
5efbfbe4d1272	hyperledger/fabric-peer:1.4.0				"peer node start" 20
minutes ago	Up 20 minutes	0.0.0.0:23051->7051/tcp, 0.0.0.0:23053->7053/tcp			
cvgalicia.colegiosveterinarios.netcan.com					
84de82313932	hyperledger/fabric-peer:1.4.0				"peer node start"
20 minutes ago	Up 20 minutes	0.0.0.0:12051->7051/tcp, 0.0.0.0:12053->7053/tcp			
cvandalucia.colegiosveterinarios.netcan.com					
b65b2075c00a	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:15984->5984/tcp	couchdb10		
16b300ca4219	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:17984->5984/tcp	couchdb12		
62cb45063c0d	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:18984->5984/tcp	couchdb13		
09283c58a6b9	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:14984->5984/tcp	couchdb9		
5133bdb07abf	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:16984->5984/tcp	couchdb11		
40eb43a49b91	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:12984->5984/tcp	couchdb7		
a46e2961fe0b	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:19984->5984/tcp	couchdb14		
a3785bbac749	hyperledger/fabric-ca:1.4.0				"sh -c 'fabric-ca-se...'"
20 minutes ago	Up 20 minutes	0.0.0.0:7054->7054/tcp	ca_federaciones		
8607f1b75ccb	hyperledger/fabric-ca:1.4.0				"sh -c 'fabric-ca-se...'"
20 minutes ago	Up 20 minutes	0.0.0.0:8054->7054/tcp	ca_colegiosveterinarios		
279fd7a1ce36	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:9984->5984/tcp	couchdb4		
acb5fe15bd11	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp	couchdb2		
88405564967c	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:13984->5984/tcp	couchdb8		
36b97918a0ba	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:24984->5984/tcp	couchdb19		
692c3207ca92	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:23984->5984/tcp	couchdb18		
fc096cd340a9	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:22984->5984/tcp	couchdb17		
6dc3acfc6566	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:25984->5984/tcp	couchdb20		
f675298d9f1d	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp	couchdb0		
4ba6d4aae8b9	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp	couchdb1		
e4789a01413e	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:20984->5984/tcp	couchdb15		
4e6b0db891c8	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-
ent..." 20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:21984->5984/tcp	couchdb16		
e38a6faa457d	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:11984->5984/tcp	couchdb6		
1e4e76f0147f	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:10984->5984/tcp	couchdb5		
90c247abc1f7	hyperledger/fabric-couchdb:0.4.15				"tini -- /docker-ent..."
20 minutes ago	Up 20 minutes	4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp	couchdb3		
349d2601803d	hyperledger/fabric-orderer:1.4.0				"orderer" 20
minutes ago	Up 20 minutes	0.0.0.0:7050->7050/tcp	orderer.netcan.com		



También podemos comprobar que se ha creado la red `tfm_netcan` que se definía en los archivos `docker-compose-base.yaml` y `peer-base.yaml`, en las variables `CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE`, utilizando el comando:

`Docker network ls`

```
hyperledger@localhost:~/work/src/fabric-samples/TFM$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
53a038bc3615    bridge    bridge      local
ea3f5cc505fa    host      host       local
619f53f028c1    none     null       local
19b1f0c96b46    tfm_netcan    bridge      local
```

Una vez levantada la red y comprobado que está funcionando correctamente podemos entrar a la CA para comprobar si está funcionando correctamente.

Para ello primero entramos a los logs de la CA para ver si está corriendo correctamente:

`docker logs -f ca_federaciones`

```
2020/09/12 12:37:06 [DEBUG] CA initialization successful
2020/09/12 12:37:06 [DEBUG] Initializing Idemix issuer...
2020/09/12 12:37:07 [INFO] The issuer key was successfully stored. The public key is at: /etc/hyperledger/fabric-ca-server/IssuerPublicKey, secret key is at: /etc/hyperledger/fabric-ca-server/msp/keystore/IssuerSecretKey
2020/09/12 12:37:07 [DEBUG] Initializing revocation authority for issuer 'ca-federaciones'
2020/09/12 12:37:07 [DEBUG] Initialize Idemix issuer revocation key material
2020/09/12 12:37:07 [INFO] Idemix issuer revocation public and secret keys were generated for CA 'ca-federaciones'
2020/09/12 12:37:07 [INFO] The revocation key was successfully stored. The public key is at: /etc/hyperledger/fabric-ca-server/IssuerRevocationPublicKey, private key is at: /etc/hyperledger/fabric-ca-server/msp/keystore/IssuerRevocationPrivateKey
2020/09/12 12:37:07 [DEBUG] Initializing nonce manager for issuer 'ca-federaciones'
2020/09/12 12:37:07 [INFO] Home directory for default CA: /etc/hyperledger/fabric-ca-server
2020/09/12 12:37:07 [DEBUG] 1 CA instance(s) running on server
2020/09/12 12:37:07 [INFO] Listening on http://0.0.0.0:7054
```

Abrimos una nueva consola, entramos al docker de la CA y nos logamos con el usuario admin de la CA mediante los comandos:

```
docker exec -it ca_federaciones bash
cd etc/hyperledger/fabric-ca-server
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
```

```
hyperledger@localhost:~/work/src/fabric-samples/TFM$ docker exec -it ca_federaciones bash
root@a3785bbac749:/# cd etc/hyperledger/fabric-ca-server
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
2020/09/12 13:10:02 [INFO] Created a default configuration file at /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml
2020/09/12 13:10:02 [INFO] generating key: &{A:ecdsa S:256}
2020/09/12 13:10:02 [INFO] encoded CSR
2020/09/12 13:10:02 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem
2020/09/12 13:10:02 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem
2020/09/12 13:10:02 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey
2020/09/12 13:10:02 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey
root@a3785bbac749:/etc/hyperledger/fabric-ca-server#
```



Una vez logados creamos un nuevo usuario, pero antes hay que crear la afiliación si no existe

```
fabric-ca-client affiliation add federaciones
```

```
fabric-ca-client affiliation add federaciones.RSCE
```

```
fabric-ca-client register --id.name Daniel --id.secret 22349 --id.affiliation federaciones.RSCE --id.attrs  
'hf.Revoker=true,admin=true:ecert'
```

```
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client affiliation add federaciones  
2020/09/12 13:11:10 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Successfully added affiliation: federaciones  
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client affiliation add federaciones.RSCE  
2020/09/12 13:11:17 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Successfully added affiliation: federaciones.RSCE  
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client register --id.name Daniel --id.secret 22349 --id.affiliation federaciones.RSCE --id.attrs 'hf.Revoker=true,admin=true:ecert'  
2020/09/12 13:11:23 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Password: 22349
```

Comprobamos que esté registrado

```
fabric-ca-client enroll -u http://Daniel:22349@localhost:7054 -M msp
```

Y volvemos al usuario admin para revocar la identidad creada:

```
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
```

```
fabric-ca-client identity remove Daniel
```

```
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://Daniel:22349@localhost:7054 -M msp  
2020/09/12 13:13:43 [INFO] generating key: &{A:ecdsa S:256}  
2020/09/12 13:13:43 [INFO] encoded CSR  
2020/09/12 13:13:43 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem  
2020/09/12 13:13:43 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem  
2020/09/12 13:13:43 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey  
2020/09/12 13:13:43 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey  
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://admin:adminpw@localhost:7054  
2020/09/12 13:13:52 [INFO] generating key: &{A:ecdsa S:256}  
2020/09/12 13:13:52 [INFO] encoded CSR  
2020/09/12 13:13:52 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem  
2020/09/12 13:13:52 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem  
2020/09/12 13:13:52 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey  
2020/09/12 13:13:52 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey  
root@a3785bbac749:/etc/hyperledger/fabric-ca-server# fabric-ca-client identity remove Daniel  
2020/09/12 13:13:59 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Successfully removed identity - Name: Daniel, Type: client, Affiliation: federaciones.RSCE, Max Enrollments: -1, Attributes: [{Name:hf.Revoker Value:true ECert:false} {Name:admin Value:true ECert:true} {Name:hf.EnrollmentID Value:Daniel ECert:true} {Name:hf.Type Value:client ECert:true} {Name:hf.Affiliation Value:federaciones.RSCE ECert:true}]
```



INSTALACIÓN DE HYPERLEDGER EXPLORER

Para poder visualizar la información de la red en el navegador instalaremos Hyperledger Explorer, que nos permitirá ver los datos de la red, los bloques creados, las transacciones y los chaincodes.

Desde la carpeta `/home/hyperledger/work/src` instalaremos los prerequisitos de Hyperledger Fabric. En particular estaremos atentos a las versiones ya que hasta el momento hemos utilizado Fabric 1.4.0, comprobando en las Release Notes de Hyperledger Fabric Explorer que deberemos utilizar la versión 8.11.1 de NodeJS y la versión 0.3.9.3 de Hyperledger Fabric Explorer.

Release Notes

Hyperledger Explorer Version	Fabric Version Supported	NodeJS Version Supported
v1.1.2 (Aug 12, 2020)	v1.4.0 to v2.2.0	12.16.x
v1.1.1 (Jul 17, 2020)	v1.4.0 to v2.1.1	12.16.x
v1.1.0 (Jul 01, 2020)	v1.4.0 to v2.1.1	12.16.x
v1.0.0 (Apr 09, 2020)	v1.4.0 to v1.4.8	10.19.x
v1.0.0-rc3 (Apr 01, 2020)	v1.4.0 to v1.4.6	10.19.x
v1.0.0-rc2 (Dec 10, 2019)	v1.4.0 to v1.4.4	8.11.x
v1.0.0-rc1 (Nov 18, 2019)	v1.4.2	8.11.x
v0.3.9.5 (Sep 8, 2019)	v1.4.2	8.11.x
v0.3.9.4 (June 18, 2019)	v1.4.1	8.11.x
v0.3.9.3 (May 24, 2019)	v1.4	8.11.x
v0.3.9.2 (May 16, 2019)	v1.4	8.11.x
v0.3.9.1 (Feb 28, 2019)	v1.4	8.11.x
v0.3.9 (Feb 7, 2019)	v1.4	8.11.x
v0.3.8 (Dec 13, 2018)	v1.3	8.x.x



Partimos de la carpeta /home/hyperledger/work/src

```
cd ~/work/src
```

E instalamos los prerequisitos (jq y NodeJS), pasando a la versión 8.11.1 de NodeJS.

```
sudo apt-get update
sudo apt-get install jq
nvm install 8.11.1
nvm use 8.11.1
```

A continuación es necesario instalar Postgresql ya que es la base de datos que utiliza Hyperledger Explorer, creando el usuario hyperledger

```
sudo apt-get install postgresql postgresql-contrib
```

Para conectarse a la consola utilizaremos:

```
sudo -u postgres psql // Para conectarse. También psql -h localhost -U hyperledger -d fabricexplorer
```

Y una vez dentro crearemos el usuario hyperledger y saldremos de la consola:

```
CREATE USER hyperledger PASSWORD 'hyperledger';
\q
```

Una vez configurado postresql descargaremos el repositorio de Hyperledger Explorer y utilizaremos la versión 0.3.9.3:

```
git clone https://github.com/hyperledger/blockchain-explorer.git
cd blockchain-explorer
git checkout v0.3.9.3
```

Con Hyperledger Explorer ya instalado podemos pasar a configurarlo. Para ello hay que modificar los siguientes ficheros, que pueden encontrarse en el archivo Configuracion_explorer.tar de la carpeta Arquitectura_bolckchain/Configuracion_explorer del repositorio Github https://github.com/DFLBB/TFM_archs:



- ./app/explorerconfig.json, en el que tendremos que indicar el usuario y la contraseña establecidos en la base de datos postgresql

```
- {
  "persistence": "postgreSQL",
  "platforms": ["fabric"],
  "postgreSQL": {
    "host": "127.0.0.1",
    "port": "5432",
    "database": "fabricexplorer",
    "username": "hyperledger",
    "passwd": "hyperledger"
  },
  "sync": {
    "type": "local",
    "platform": "fabric",
    "blocksSyncTime": "1"
  },
  "jwt": {
    "secret": "a secret phrase!!",
    "expiresIn": "2h"
  }
}
```

- ./app/platform/fabric/config.json, en donde hay que indicar el nombre de la red y el archivo de configuración

```
{
  "network-configs": {
    "netcan": {
      "name": "netcan",
      "profile": "./connection-profile/netcan_15.json"
    }
  },
  "license": "Apache-2.0"
}
```

- ./app/platform/fabric/connection-profile/netcan.json, donde definiremos la red, la organización y el canal al que se conectará el Hyperledger Explorer

```
- {
  "name": "netcan",
  "version": "1.0.0",
  "license": "Apache-2.0",
  "client": {
    "tlsEnable": false,
    "adminUser": "admin",
    "adminPassword": "adminpw",
    "enableAuthentication": false,
    "organization": "Federaciones",
    "connection": {
      "timeout": {
        "peer": {
          "endorser": "300",
          "orderer": "300"
        }
      }
    },
    "channels": {
      "netcanchannel": {
        "peers": {
          "fci.federaciones.netcan.com": {}
        },
        "connection": {
          "timeout": {
            "peer": {
              "endorser": "300",
              "orderer": "300"
            }
          }
        }
      }
    }
  }
}
```



```
-          "endorser": "6000",
-          "eventHub": "6000",
-          "eventReg": "6000"
-      }
-    }
-  }
},
"organizations": {
  "Federaciones": {
    "mspid": "FederacionesMSP",
    "fullpath": true
  }
},
"peers": {
  "fci.federaciones.netcan.com": {
    "url": "grpc://localhost:7051",
    "eventUrl": "grpc://localhost:7053",
    "grpcOptions": {
      "ssl-target-name-override": "fci.federaciones.netcan.com"
    }
  }
}
}
```

Además estableceremos las variables de entorno ya que a veces dan problemas:

```
export DATABASE_HOST=127.0.0.1
export DATABASE_PORT=5432
export DATABASE_DATABASE=fabricexplorer
export DATABASE_USERNAME=hyperledger
export DATABASE_PASSWORD=hyperledger
```

A continuación crearemos la base de datos de Hyperledger Explorer en postgresql:

```
cd /home/hyperledger/work/src/blockchain-explorer/app/persistence/fabric/postgreSQL/
chmod -R 775 db
cd db/
./createdb.sh
```

Y por último ejecutaremos el script que instala Hyperledger Explorer:

```
cd /home/hyperledger/work/src/blockchain-explorer/
./main.sh install
```

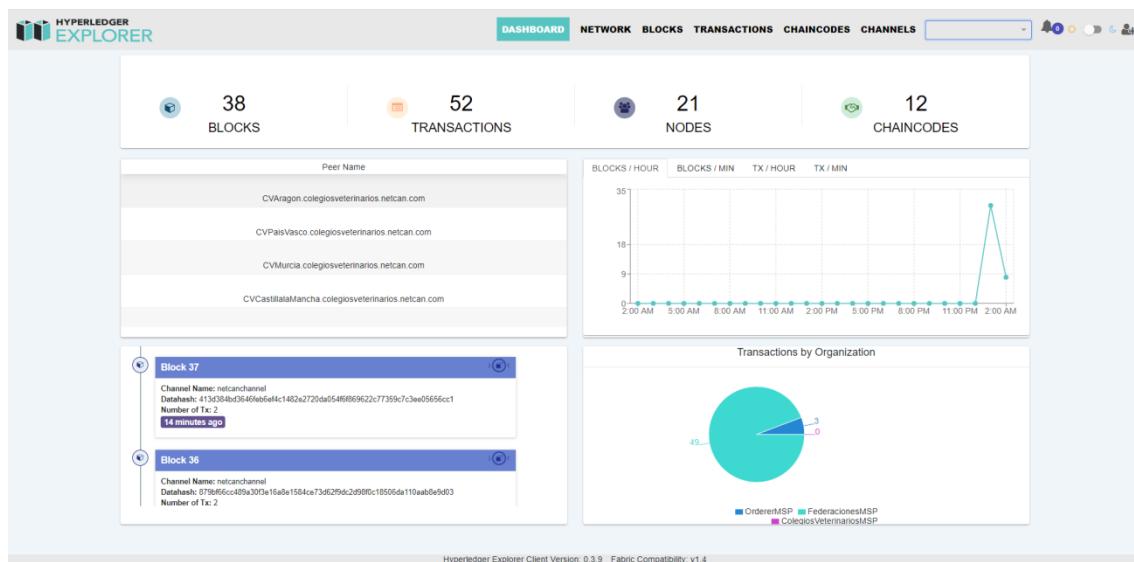
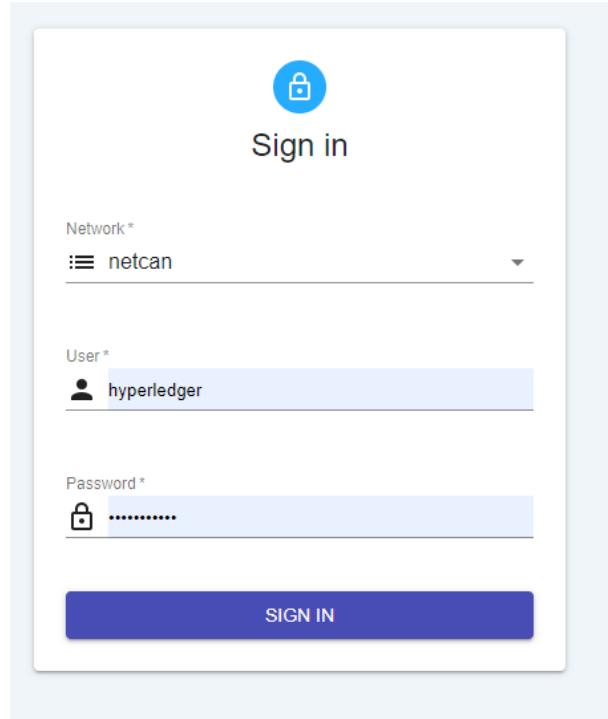
Con la red levantada se utiliza el script start para inicial el servicio

```
./start.sh
```



Y podremos ver los logs en las carpetas app, console y db situadas en /home/hyperledger/work/src/blockchain-explorer/logs/

Por último se puede acceder en el navegador en el puerto :8080, en donde podemos observar el resumen de la actividad:





La actividad en la red:

The screenshot shows the 'NETWORK' tab of the Hyperledger Explorer interface. It displays a table of peers and their status. The columns include Peer Name, Request Url, Peer Type, MSPID, and Ledger Height (High, Low, Unsigned). The table lists five peers: CVAragon.colegiosveterinario, CVPaisVasco.colegiosveterinario, CVMurcia.colegiosveterinarios, CVCastillalaMancha.colegiosveterinario, and CVNavarra.colegiosveterinario. All peers are listed as PEER type with ColegiosVeterinariosMSP as the MSPID. The ledger height for all is 0, with High and Low both at 38, and Unsigned set to true. At the bottom, there are navigation links for Previous, Page 1 of 5, 5 rows, and Next.

Los bloques creados:

The screenshot shows the 'BLOCKS' tab of the Hyperledger Explorer interface. It displays a table of blocks. The columns include From, To, Select Orgs, Search, Reset, Clear Filter, Block Number, Channel Name, Number of Tx, Data Hash, Block Hash, Previous Hash, and Transactions. The table lists seven blocks from September 9, 2020, to September 10, 2020. Each block has a unique number, channel name (netcanchannel), transaction count (2 or 1), data hash, block hash, previous hash, and a long string of characters representing the transactions. At the bottom, there are navigation links for Previous, Page 1 of 4, 10 rows, and Next.



Las transacciones:

The screenshot shows the Hyperledger Explorer interface with the 'TRANSACTIONS' tab selected. The table lists ten transactions, all of which are 'ENDORSER_TRANSACTION' type. The 'Creator' column shows 'FederacionesMSP'. The 'Channel Name' column shows 'netcchannel'. The 'Tx Id' column contains various transaction IDs. The 'Type' column shows 'ENDORSER_TRANSACTION'. The 'Chaincode' column shows categories like 'vacunas' and 'microchips'. The 'Timestamp' column shows dates ranging from 2020-09-10T00:01:25.292Z to 2020-09-10T00:01:17.106Z. The bottom of the page indicates 'Hyperledger Explorer Client Version: 0.3.9 - Fabric Compatibility: v1.4'.

From	To	Creator	Channel Name	Tx Id	Type	Chaincode	Timestamp
September 9, 2020 2:17 AM	September 10, 2020 2:17 AM	FederacionesMSP	netcchannel	9c0607...	ENDORSER_TRANSACTION	vacunas	2020-09-10T00:01:25.292Z
		FederacionesMSP	netcchannel	e5004f...	ENDORSER_TRANSACTION	vacunas	2020-09-10T00:01:25.292Z
		FederacionesMSP	netcchannel	64d2af...	ENDORSER_TRANSACTION	vacunas	2020-09-10T00:01:17.106Z
		FederacionesMSP	netcchannel	a30615...	ENDORSER_TRANSACTION	vacunas	2020-09-10T00:01:17.106Z
		FederacionesMSP	netcchannel	7fa4e2...	ENDORSER_TRANSACTION	vacunas	2020-09-10T00:01:08.402Z
		FederacionesMSP	netcchannel	6ba1bd...	ENDORSER_TRANSACTION	microchips	2020-09-10T00:01:00.241Z
		FederacionesMSP	netcchannel	d65226...	ENDORSER_TRANSACTION	microchips	2020-09-10T00:00:38.184Z
		FederacionesMSP	netcchannel	63a650...	ENDORSER_TRANSACTION	microchips	2020-09-10T00:00:18.798Z
		FederacionesMSP	netcchannel	db082c...	ENDORSER_TRANSACTION	microchips	2020-09-10T00:00:01.440Z
		FederacionesMSP	netcchannel	hh4nrt...	ENDORSER_TRANSACTION	microchips	2020-09-10T23:49:50.203Z

Y la actividad de los chaincode:

The screenshot shows the Hyperledger Explorer interface with the 'CHAINCODES' tab selected. The table lists five chaincodes: 'afios', 'drm', 'marbles', 'microchips', and 'perfles'. Each entry includes the 'Chaincode Name', 'Channel Name', 'Path', 'Transaction Count', and 'Version'. All entries show a transaction count of 52 and a version of 1.0.0. The bottom of the page indicates 'Hyperledger Explorer Client Version: 0.3.9 - Fabric Compatibility: v1.4'.

Chaincode Name	Channel Name	Path	Transaction Count	Version
afios		github.com/chaincode/netcan/afios/cc	52	1.0.0
drm		github.com/chaincode/netcan/drm/cc	52	1.0.0
marbles		github.com/chaincode/netcan/marbles/cc	52	1.0.0
microchips		github.com/chaincode/netcan/microchips/cc	52	1.0.0
perfles		github.com/chaincode/netcan/perfles/cc	52	1.0.0



RED CON TLS

A continuación se modifica la red para implementar el protocolo TLS en el orderer, CLI y peers.

Los archivos de configuración pueden encontrarse en el repositorio Github mencionado anteriormente:

https://github.com/DFLBB/TFM_archs

En la carpeta Arquitectura_bolckchain/1_server_TLS. Los scrips se encuentran en la carpeta scripts1 del repositorio.

Se trata de una copia completa del proyecto.

El procedimiento realizado es el mismo y los archivos a definir también, tratándose realmente de la red definida en el apartado 2.5 de este documento, por lo que no repetiremos los pasos a realizar.

Únicamente varían con respecto a la red anterior los archivos docker-compose-cli.yaml, docker-compose-base.yaml y peer-base.yaml en los que debe introducirse todo lo referente al TLS tal y como se explicó en el apartado 2.5, es decir, definir correctamente las siguientes variables:

- CORE_PEER_TLS_ENABLED=true
- CORE_PEER_TLS_CERT_FILE, CORE_PEER_TLS_KEY_FILE y CORE_PEER_TLS_ROOTCERT_FILE
- ORDERER_GENERAL_TLS_PRIVATEKEY, ORDERER_GENERAL_TLS_CETIFICATE y ORDERER_GENERAL_TLS_ROOTCAS

El resultado final tras lanzar el script de inicio es el mismo:

```
*****
Cargando datos de VACUNACIONES
*****
2020-09-12 16:24:00.933 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"docType\":\"CONTADOR\",\"IDMaximo\":11}"
2020-09-12 16:24:01.184 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"89"
2020-09-12 16:24:08.342 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"docType\":\"CONTADOR\",\"IDMaximo\":89}"
2020-09-12 16:24:09.001 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"514"
2020-09-12 16:24:16.162 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:200 payload:"{\"docType\":\"CONTADOR\",\"IDMaximo\":512}"
*****
DATOS INICIALES CARGADOS
*****
```



Podemos comprobar también que los dockers están funcionando correctamente:

```
hyperledger@localhost:~/work/src/fabric-samples/TFM2$ docker ps -a
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              NAMES
a586c944e147        dev-fci.federaciones.netcan.com-veterinarios-1.0.0-
130c44c62ff0205de3b79eea6e6c6fdb54e461931866f81fb4bf3ad7300101fd   "chaincode -peer.add..." 7 minutes ago   Up 7
minutes
f3195d77e341        dev-fci.federaciones.netcan.com-veterinarios-1.0.0-
c0712aaa2c66f0eb4d3e68b9dca83fb740be117b8e8c9e54b50a94da72f3f366   "chaincode -peer.add..." 7 minutes ago
Up 7 minutes
6faaa4b9c88b3        dev-fci.federaciones.netcan.com-vacunas-1.0.0-
de8c71563a9bf41548efc3efd696676c22d66525e1f312b37574d60c1b17aab0   "chaincode -peer.add..." 8 minutes ago   Up
8 minutes
c25afd673a69        dev-fci.federaciones.netcan.com-solicitudes-1.0.0-
b5e08411bd92c32ee9f31c54d803da0da9dbc07c35b86bdd9dbc66c7d9379d6c   "chaincode -peer.add..." 9 minutes ago
Up 9 minutes
d028e389859a        dev-fci.federaciones.netcan.com-razas-1.0.0-
ec9fd3c3016785411df3b0cfb73fb0c4a2ff8f249ada3ae8604d52579cf2ca92   "chaincode -peer.add..." 9 minutes ago   Up
9 minutes
bf5699f9e24d        dev-fci.federaciones.netcan.com-perros-1.0.0-
46b41e94180f7b7b9d3a25abd665310a9fe68ed5639e9ab474ddf30c60d278f3   "chaincode -peer.add..." 10 minutes ago
Up 10 minutes
4590c88cacba       dev-fci.federaciones.netcan.com-perfiles-1.0.0-
ee62113a5af70130e6a0df3310ec7555afcaee7f3659eb7a4191efdf56bfa5d8   "chaincode -peer.add..." 10 minutes ago
Up 10 minutes
55601d19cebb       dev-fci.federaciones.netcan.com-microchips-1.0.0-
fb61a6a31883fa2a1b3280a342942d5462730c2b4264e4e39e8f7cec9785092c   "chaincode -peer.add..." 11 minutes ago
Up 11 minutes
988d7da3eb9b       dev-fci.federaciones.netcan.com-afijos-1.0.0-
33d55e901eadaa70caea32134c737a7fc4caa1ca66c9f8a937444c5383c8dbe   "chaincode -peer.add..." 11 minutes ago
Up 11 minutes
f7d350837a53        hyperledger/fabric-tools:1.4.0           "/bin/bash"          13
minutes ago   Up 13 minutes
98607d432073       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvextremadura.colegiosveterinarios.netcan.com
9002d51c0353       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes 0.0.0.0:10051->7051/tcp, 0.0.0.0:10053->7053/tcp acw.federaciones.netcan.com
b3c24b94564f       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvcnarias.colegiosveterinarios.netcan.com
874859966c08       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvcataluna.colegiosveterinarios.netcan.com
9bd477f12e2e       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvcantabria.colegiosveterinarios.netcan.com
1be18d94cee9       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvillesbalears.colegiosveterinarios.netcan.com
1c342f976a4a       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvnavarra.colegiosveterinarios.netcan.com
de6e006ec6af       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvcastillayleon.colegiosveterinarios.netcan.com
0d33b90f342c       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvlarioja.colegiosveterinarios.netcan.com
44f70682c94b       hyperledger/fabric-peer:1.4.0            "peer node start"
13 minutes ago   Up 13 minutes
cvmurcia.colegiosveterinarios.netcan.com
```



aef5b718652f	hyperledger/fabric-peer:1.4.0	"peer node start" 13
minutes ago	Up 13 minutes	0.0.0.0:14051->7051/tcp, 0.0.0.0:14053->7053/tcp
cvesturias.colegiosveterinarios.netcan.com		
7f4edd208e32	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:23051->7051/tcp, 0.0.0.0:23053->7053/tcp
cvgalicia.colegiosveterinarios.netcan.com		
32456cd98b79	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:26051->7051/tcp, 0.0.0.0:26053->7053/tcp
cpaisvasco.colegiosveterinarios.netcan.com		
99ca8c595e72	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:7051->7051/tcp, 0.0.0.0:7053->7053/tcp fci.federaciones.netcan.com
e345ef4c99e1	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:19051->7051/tcp, 0.0.0.0:19053->7053/tcp
cvcastillalamancha.colegiosveterinarios.netcan.com		
bf74300b19e0	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:21051->7051/tcp, 0.0.0.0:21053->7053/tcp
cvcomunitatvalenciana.colegiosveterinarios.netcan.com		
372e2dd9a544	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:9051->7051/tcp, 0.0.0.0:9053->7053/tcp tkc.federaciones.netcan.com
8976910066ce	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:11051->7051/tcp, 0.0.0.0:11053->7053/tcp
cmadrid.colegiosveterinarios.netcan.com		
782af996da0b	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:8051->7051/tcp, 0.0.0.0:8053->7053/tcp rsce.federaciones.netcan.com
962cefc6f9d3	hyperledger/fabric-peer:1.4.0	"peer node start" 13
minutes ago	Up 13 minutes	0.0.0.0:12051->7051/tcp, 0.0.0.0:12053->7053/tcp
cvandalucia.colegiosveterinarios.netcan.com		
88d2c08cbfd	hyperledger/fabric-peer:1.4.0	"peer node start"
13 minutes ago	Up 13 minutes	0.0.0.0:13051->7051/tcp, 0.0.0.0:13053->7053/tcp
cvaragon.colegiosveterinarios.netcan.com		
ea6d64f28ccc	hyperledger/fabric-ca:1.4.0	"sh -c 'fabric-ca-se...'"
13 minutes ago	Up 13 minutes	0.0.0.0:8054->7054/tcp ca_colegiosveterinarios
aad365324580	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:14984->5984/tcp couchdb9
a71ce0ce9753	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:7984->5984/tcp couchdb2
7463ccf0857a	hyperledger/fabric-orderer:1.4.0	"orderer" 13
minutes ago	Up 13 minutes	0.0.0.0:7050->7050/tcp orderer.netcan.com
19ec5a46ae2b	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:22984->5984/tcp couchdb17
53c7c6a4cc9e	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:8984->5984/tcp couchdb3
8fbe12607240	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:25984->5984/tcp couchdb20
a7b3ef2ba8cc	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:23984->5984/tcp couchdb18
00c7f01324bc	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:20984->5984/tcp couchdb15
458978aac3ec	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:18984->5984/tcp couchdb13
77768de52068	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:12984->5984/tcp couchdb7
3b13146d8511	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:21984->5984/tcp couchdb16
9c203052f7be	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:17984->5984/tcp couchdb12
73324fd3a54e	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:16984->5984/tcp couchdb11
295b8c6d4dd3	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:5984->5984/tcp couchdb0
c2731bd80cea	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-
ent..." 13 minutes ago	Up 13 minutes	4369/tcp, 9100/tcp, 0.0.0.0:9984->5984/tcp couchdb4
3fb914af4804	hyperledger/fabric-ca:1.4.0	"sh -c 'fabric-ca-se...'"
13 minutes ago	Up 13 minutes	0.0.0.0:7054->7054/tcp ca_federaciones



499058f050c7	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:11984->5984/tcp	couchdb6
1e979f2fd299	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:19984->5984/tcp	couchdb14
7d6f86c8b9cb	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:24984->5984/tcp	couchdb19
7caf66b62cba	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:15984->5984/tcp	couchdb10
1fca5ebefddc	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:6984->5984/tcp	couchdb1
764fbcc1e1eb7	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:13984->5984/tcp	couchdb8
cfaf325c3109	hyperledger/fabric-couchdb:0.4.15	"tini -- /docker-ent..."
13 minutes ago	Up 13 minutes 4369/tcp, 9100/tcp, 0.0.0.0:10984->5984/tcp	couchdb5

Y podemos igualmente lograrnos a la CA y crear nuevas afiliaciones:

```
docker exec -it ca_federaciones bash
cd etc/hyperledger/fabric-ca-server
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
fabric-ca-client affiliation add federaciones
fabric-ca-client affiliation add federaciones.RSCE
```

```
hyperledger@localhost:~/work/src/fabric-samples/TFM2$ docker exec -it ca_federaciones bash
root@3fb914af4804:/# cd etc/hyperledger/fabric-ca-server
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
2020/09/12 16:33:43 [INFO] Created a default configuration file at /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml
2020/09/12 16:33:43 [INFO] generating key: &{A:ecdsa S:256}
2020/09/12 16:33:43 [INFO] encoded CSR
2020/09/12 16:33:43 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem
2020/09/12 16:33:43 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem
2020/09/12 16:33:43 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey
2020/09/12 16:33:43 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client affiliation add federaciones
2020/09/12 16:33:43 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml
Successfully added affiliation: federaciones
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client affiliation add federaciones.RSCE
2020/09/12 16:33:43 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml
Successfully added affiliation: federaciones.RSCE
```

Crear un nuevo usuario:

```
fabric-ca-client register --id.name Unai --id.secret 99876 --id.affiliation federaciones.RSCE --id.attrs
'hf.Revoker=true,admin=true:ecert'
fabric-ca-client enroll -u http://Unai:99876@localhost:7054 -M msp
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
```



```
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client register --id.name Unai --id.secret 99876 --id.affiliation federaciones.RSCE --id.attrs 'hf.Revoker=true,admin=true,ecert'  
2020/09/12 16:34:25 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Password: 99876  
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://Unai:99876@localhost:7054 -M msp  
2020/09/12 16:34:37 [INFO] generating key: &{A:ecdsa S:256}  
2020/09/12 16:34:37 [INFO] encoded CSR  
2020/09/12 16:34:37 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem  
2020/09/12 16:34:37 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem  
2020/09/12 16:34:37 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey  
2020/09/12 16:34:37 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey  
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client enroll -u http://admin:adminpw@localhost:7054  
2020/09/12 16:34:43 [INFO] generating key: &{A:ecdsa S:256}  
2020/09/12 16:34:43 [INFO] encoded CSR  
2020/09/12 16:34:43 [INFO] Stored client certificate at /etc/hyperledger/fabric-ca-server/msp/signcerts/cert.pem  
2020/09/12 16:34:43 [INFO] Stored root CA certificate at /etc/hyperledger/fabric-ca-server/msp/cacerts/localhost-7054.pem  
2020/09/12 16:34:43 [INFO] Stored Issuer public key at /etc/hyperledger/fabric-ca-server/msp/IssuerPublicKey  
2020/09/12 16:34:43 [INFO] Stored Issuer revocation public key at /etc/hyperledger/fabric-ca-server/msp/IssuerRevocationPublicKey
```

Y borrarlo:

```
fabric-ca-client identity remove Unai
```

```
root@3fb914af4804:/etc/hyperledger/fabric-ca-server# fabric-ca-client identity remove Unai  
2020/09/12 16:35:42 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-client-config.yaml  
Successfully removed identity - Name: Unai, Type: client, Affiliation: federaciones.RSCE, Max Enrollments: -1, Attributes: [{Name:hf.Type Value:client ECert:true} {Name:hf.Affiliation Value:federaciones.RSCE ECert:true} {Name:hf.Revoker Value:true ECert:false} {Name:admin Value:true ECert:true} {Name:hf.EnrollmentID Value:Unai ECert:true}]  
root@3fb914af4804:/etc/hyperledger/fabric-ca-server#
```



RED CREADA ENTRE DOS SERVIDORES

Por último se implementa una red más parecida a la realidad en la que cada organización se despliega en un servidor distinto, utilizándose una red Docker Swarm para comunicar ambos servidores.

Los archivos pueden encontrarse en el repositorio Github en la carpeta Arquitectura_Arquitectura_bolckchain/2_servers_TLS y los scripts en scripts3.

- Creación de la red Docker Swarm

En el primer servidor, que en nuestro caso será el servidor de Federaciones, en la carpeta del proyecto (que en nuestro caso es TFM), buscamos en primer lugar la interfaz de red de la IP pública del servidor:

ifconfig

```
root@localhost:~# ifconfig
docker0  Link encap:Ethernet HWaddr 02:42:3c:91:bb:c2
        inet addr:172.17.0.1 Bcast:172.17.255.255 Mask:255.255.0.0
          inet6 addr: fe80::42:3cff:fe91:bbc2/64 Scope:Link
            UP BROADCAST MULTICAST MTU:1500 Metric:1
            RX packets:0 errors:0 dropped:0 overruns:0 frame:0
            TX packets:69 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:0 (0.0 B) TX bytes:6198 (6.1 KB)

docker_gwbridge Link encap:Ethernet HWaddr 02:42:47:7a:68:06
        inet addr:192.168.160.1 Bcast:192.168.175.255 Mask:255.255.240.0
          inet6 addr: fe80::42:47ff:fe7a:6806/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:16914 errors:0 dropped:0 overruns:0 frame:0
            TX packets:17903 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:0
            RX bytes:2089040 (2.0 MB) TX bytes:1867934 (1.8 MB)

ens192  Link encap:Ethernet HWaddr 00:50:56:3b:30:09
        inet addr:82.223.122.137 Bcast:82.223.122.137 Mask:255.255.255.255
          inet6 addr: fe80::250:56ff:fe3b:3009/64 Scope:Link
            UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
            RX packets:45509944 errors:0 dropped:0 overruns:0 frame:0
            TX packets:44016475 errors:0 dropped:0 overruns:0 carrier:0
            collisions:0 txqueuelen:1000
            RX bytes:11207888057 (11.2 GB) TX bytes:8919839812 (8.9 GB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1 Mask:255.0.0.0
```

A continuación creamos una red Docker Swarm y le indicamos la interfaz de red de la IP pública

docker swarm init --advertise-addr ens192

```
root@localhost:/home/hyperledger/work/src/fabric-samples# docker swarm init --advertise-addr ens192
Swarm initialized: current node (3n2b1hs0uv9nospftt02b80qtm) is now a manager.
```

To add a worker to this swarm, run the following command:

```
docker swarm join --token SWMTKN-1-62wv8gfk5g030i9rgv2ytnqpg4mazxyvm3tyzu51q616jppsto-746cpa6lvaapaf9foj3rhj64f
82.223.122.137:2377
```

To add a manager to this swarm, run 'docker swarm join-token manager' and follow the instructions.

Como en el segundo servidor queremos tener también un nodo manager utilizamos la siguiente instrucción:

```
docker swarm join-token manager
```

Que genera un token que pegaremos en la consola del segundo servidor, en el que estaremos en la carpeta del proyecto:

```
root@localhost:/home/hyperledger/work/src/fabric-samples# docker swarm join-token manager
To add a manager to this swarm, run the following command:
```

```
docker swarm join --token SWMTKN-1-62wv8gfk5g030i9rgv2ytnqpg4mazxyvm3tyzu51q616jppsto-418znypmnts mx9az6ivzjco04 82.223.122.137:2377
```

Una vez realizado esto regresamos al primer servidor en el que arrancaremos la red con el siguiente comando:

```
docker network create --attachable --driver overlay netcan
```

Pudiendo comprobar con

```
docker network ls
```

Que en ambos servidores aparece la misma red.

```
11 82.223.122.137 (Federaciones-J) 12.82.223.101.251 (CVeterinarios-C)
root@localhost:/home/hyperledger/work/src/fabric-samples/TFM# docker network ls
NETWORK ID      NAME        DRIVER      SCOPE
991d0bf6967f   bridge      bridge      local
8751f78be7a2   docker_gwbridge  bridge      local
548a6ed22363   host        host       local
owqyfz8d0gw1   ingress     overlay    swarm
7n5k7vaewf1o   netcan      overlay    swarm
b3e3t7c14dc6   none        null      local

11 82.223.122.137 (Federaciones-J) 12.82.223.101.251 (CVeterinarios-C)
root@localhost:/home/hyperledger/work/src/fabric-samples/TFM# docker network ls
NETWORK ID      NAME        DRIVER      SCOPE
64cd714d221f   bridge      bridge      local
771fad0626dd   docker_gwbridge  bridge      local
00e0477a2d27   host        host       local
owqyfz8d0gw1   ingress     overlay    swarm
7n5k7vaewf1o   netcan      overlay    swarm
ab97857c695e   none        null      local
```

- Configuración de la red

En primer lugar, dado que queremos seguir manteniendo los scripts que levantan la red, necesitamos un método para poder conectar por medio de un script desde el primer servidor con la consola del segundo servidor y lanzar allí los comandos correspondientes, lo que realizamos configurando una clave pública/privada con ssh-keygen:



Generamos el par de claves mediante el comando:

```
ssh-keygen -b 4096 -t rsa
```

```
root@localhost:/home/hyperledger/work/src/fabric-samples/TFM# ssh-keygen -b 4096 -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:9zajio17DIKrcs2xsmOgNIlqq9BS8xFcX8i4zA8WkEk root@localhost
The key's randomart image is:
----[RSA 4096]----+
| oE=.o .. |
| = o.o. |
| + o. |
| ..o.* |
|+o+o o.oS . |
|=B.o... . . |
|O.= . o = |
|+= o= o o |
|o.... +++.. |
+----[SHA256]----+
```

Quedando la clave guardada en la carpeta .ssh del usuario en /home/hyperledger/.ssh/id_rsa

A continuación es necesario pasar la clave pública al segundo servidor mediante el comando:

```
ssh-copy-id root@82.223.101.251
```

De esta manera cuando el primer servidor tenga que ejecutar cualquier comando en el segundo servidor podrá conectarse enviando la clave y no será necesario introducir contraseñas de manera manual.

Una vez configurada la conexión entre ambos servidores procedemos a modificar los archivos de configuración de la red docker-compose-cli.yaml, docker-compose-couch.yaml y docker-compose-base.yaml, partiendo de la red generada con TLS.

En el archivo docker-compose-cli.yaml es necesario quitar para el primer servidor todo lo relativo a los peers de la organización ColegiosVeterinarios, quedando la CA de federaciones, el orderer, el cli y los peers de Federaciones.

Es importante también cambiar la definición de la red que deberá definirse como externa y con el nombre de la red Swarm, aunque en la definición de cada servicio se utilizará el nombre de la red definida como externa (default en este caso) y no el de la red Swarm:



```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

volumes:
  orderer.netcan.com:
  FCI.federaciones.netcan.com:
  RSCE.federaciones.netcan.com:
  TKC.federaciones.netcan.com:
  ACW.federaciones.netcan.com:

networks:
  default:
    external:
      name: netcan

services:

  ca.federaciones:
    image: hyperledger/fabric-ca:1.4.0
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca-federaciones
      - FABRIC_CA_SERVER_TLS_ENABLED=false
      - FABRIC_CA_SERVER_PORT=7054
    ports:
      - "7054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-config/ca.federaciones.netcan.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-config/876ab47bd6ac9132e57aa5e92828193fc70dc51522c932666e5e757a31b65de0_sk -b admin:adminpw -d --cfg.identities.allowremove'
    volumes:
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
      - /home/hyperledger/work/src/fabric-samples/TFM/crypto-
config/peerOrganizations/federaciones.netcan.com/ca:/etc/hyperledger/fabric-ca-server-config
    container_name: ca_federaciones
  networks:
    - default

  orderer.netcan.com:
    extends:
      file: base/docker-compose-base.yaml
      service: orderer.netcan.com
    container_name: orderer.netcan.com
  networks:
    - default

  FCI.federaciones.netcan.com:
    container_name: FCI.federaciones.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: FCI.federaciones.netcan.com
    networks:
      - default

  RSCE.federaciones.netcan.com:
    container_name: RSCE.federaciones.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: RSCE.federaciones.netcan.com
    networks:
      - default

  TKC.federaciones.netcan.com:
    container_name: TKC.federaciones.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: TKC.federaciones.netcan.com
    networks:
      - default

  ACW.federaciones.netcan.com:
    container_name: ACW.federaciones.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: ACW.federaciones.netcan.com
    networks:
      - default
```



```
cli:
  container_name: cli
  image: hyperledger/fabric-tools:1.4.0
  tty: true
  stdin_open: true
  environment:
    - GOPATH=/opt/gopath
    - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
    - FABRIC_LOGGING_SPEC=INFO
    - CORE_PEER_ID=cli
    - CORE_PEER_ADDRESS=FCI.federaciones.netcan.com:7051
    - CORE_PEER_LOCALMSPID=FederacionesMSP
    - CORE_PEER_TLS_ENABLED=true
    -
    CORE_PEER_TLS_CERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/FCI.federaciones.netcan.com/tls/server.crt
    -
    CORE_PEER_TLS_KEY_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/FCI.federaciones.netcan.com/tls/server.key
    -
    CORE_PEER_TLS_ROOTCERT_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/peers/FCI.federaciones.netcan.com/tls/ca.crt
    -
    CORE_PEER_MSPCONFIGPATH=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/peerOrganizations/federaciones.netcan.com/users/Admin@federaciones.netcan.com/msp
      working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
      command: /bin/bash
      volumes:
        - /var/run/:/host/var/run/
        - /home/hyperledger/work/src/fabric-samples/TFM/chaincode:/opt/gopath/src/github.com/chaincode
        - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
        - ./scripts:/opt/gopath/src/github.com/hyperledger/fabric/peer/scripts/
        - ./channel-artifacts:/opt/gopath/src/github.com/hyperledger/fabric/peer/channel-artifacts
  depends_on:
    - orderer.netcan.com
    - FCI.federaciones.netcan.com
    - RSCE.federaciones.netcan.com
    - TKC.federaciones.netcan.com
    - ACW.federaciones.netcan.com
  networks:
    - default
```

En el segundo servidor únicamente quedará lo relativo a los peers y la CA de la organización ColegiosVeterinarios:

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

volumes:
  CVAndalucia.colegiosveterinarios.netcan.com:
  CVAragon.colegiosveterinarios.netcan.com:
  CVAsturias.colegiosveterinarios.netcan.com:
  CVIllesBalears.colegiosveterinarios.netcan.com:
  CVCanarias.colegiosveterinarios.netcan.com:
  CVCantabria.colegiosveterinarios.netcan.com:
  CVCastillaLeon.colegiosveterinarios.netcan.com:
  CVCastillaLaMancha.colegiosveterinarios.netcan.com:
  CVCataluna.colegiosveterinarios.netcan.com:
  CVComunitatValenciana.colegiosveterinarios.netcan.com:
  CVExtremadura.colegiosveterinarios.netcan.com:
  CGalicia.colegiosveterinarios.netcan.com:
  CVMadrid.colegiosveterinarios.netcan.com:
  CVMurcia.colegiosveterinarios.netcan.com:
  CVNavarra.colegiosveterinarios.netcan.com:
  CPaisVasco.colegiosveterinarios.netcan.com:
  CLaRioja.colegiosveterinarios.netcan.com:

networks:
  default:
    external:
      name: netcan
```



```
services:

  ca.colegiosveterinarios:
    image: hyperledger/fabric-ca:1.4.0
    environment:
      - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
      - FABRIC_CA_SERVER_CA_NAME=ca_colegiosveterinarios
      - FABRIC_CA_SERVER_TLS_ENABLED=false
      - FABRIC_CA_SERVER_PORT=7054
    ports:
      - "8054:7054"
    command: sh -c 'fabric-ca-server start --ca.certfile /etc/hyperledger/fabric-ca-server-
config/ca_colegiosveterinarios.netcan.com-cert.pem --ca.keyfile /etc/hyperledger/fabric-ca-server-
config/1f001e10447361e790c31d0e0c0fe5b6238804142c0ec886bc2e1c8cd5a18c3336_sk -b admin-cv:admin-cvpw -d --
cfg.identities.allowremove'
    volumes:
      - ./crypto-config/peerOrganizations/colegiosveterinarios.netcan.com/ca/:/etc/hyperledger/fabric-ca-server-config
      - ./crypto-config:/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/
    container_name: ca_colegiosveterinarios
    networks:
      - default

  CVAndalucia.colegiosveterinarios.netcan.com:
    container_name: CVAndalucia.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVAndalucia.colegiosveterinarios.netcan.com
    networks:
      - default

  CVAragon.colegiosveterinarios.netcan.com:
    container_name: CVAragon.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVAragon.colegiosveterinarios.netcan.com
    networks:
      - default

  CVAsturias.colegiosveterinarios.netcan.com:
    container_name: CVAsturias.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVAsturias.colegiosveterinarios.netcan.com
    networks:
      - default

  CVIllesBalears.colegiosveterinarios.netcan.com:
    container_name: CVIllesBalears.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVIllesBalears.colegiosveterinarios.netcan.com
    networks:
      - default

  CVCanarias.colegiosveterinarios.netcan.com:
    container_name: CVCanarias.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVCanarias.colegiosveterinarios.netcan.com
    networks:
      - default

  CVCantabria.colegiosveterinarios.netcan.com:
    container_name: CVCantabria.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVCantabria.colegiosveterinarios.netcan.com
    networks:
      - default

  CVCastillayLeon.colegiosveterinarios.netcan.com:
    container_name: CVCastillayLeon.colegiosveterinarios.netcan.com
    extends:
      file: base/docker-compose-base.yaml
      service: CVCastillayLeon.colegiosveterinarios.netcan.com
    networks:
      - default

  CVCastillalaMancha.colegiosveterinarios.netcan.com:
```



```
container_name: CVCastillalaMancha.colegiosveterinarios.netcan.com
extends:
  file: base/docker-compose-base.yaml
  service: CVCastillalaMancha.colegiosveterinarios.netcan.com
networks:
  - default

CVCataluna.colegiosveterinarios.netcan.com:
  container_name: CVCataluna.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVCataluna.colegiosveterinarios.netcan.com
  networks:
    - default

CVComunitatValenciana.colegiosveterinarios.netcan.com:
  container_name: CVComunitatValenciana.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVComunitatValenciana.colegiosveterinarios.netcan.com
  networks:
    - default

CVExtremadura.colegiosveterinarios.netcan.com:
  container_name: CVExtremadura.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVExtremadura.colegiosveterinarios.netcan.com
  networks:
    - default

CVGalicia.colegiosveterinarios.netcan.com:
  container_name: CVGalicia.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVGalicia.colegiosveterinarios.netcan.com
  networks:
    - default

CVMadrid.colegiosveterinarios.netcan.com:
  container_name: CVMadrid.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVMadrid.colegiosveterinarios.netcan.com
  networks:
    - default

CVMurcia.colegiosveterinarios.netcan.com:
  container_name: CVMurcia.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVMurcia.colegiosveterinarios.netcan.com
  networks:
    - default

CVNavarra.colegiosveterinarios.netcan.com:
  container_name: CVNavarra.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVNavarra.colegiosveterinarios.netcan.com
  networks:
    - default

CVPaisVasco.colegiosveterinarios.netcan.com:
  container_name: CVPaisVasco.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVPaisVasco.colegiosveterinarios.netcan.com
  networks:
    - default

CVLaRioja.colegiosveterinarios.netcan.com:
  container_name: CVLaRioja.colegiosveterinarios.netcan.com
  extends:
    file: base/docker-compose-base.yaml
    service: CVLaRioja.colegiosveterinarios.netcan.com
  networks:
    - default
```



En el archivo docker-compose-couch.yaml en el primer servidor se dejará únicamente lo relativo a los peers de la organización Federaciones y se deberá tener cuidado en definir la red igual que en el anterior archivo:

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'
networks:
  default:
    external:
      name: netcan

services:
  couchdb0:
    container_name: couchdb0
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "5984:5984"
    networks:
      - default

  FCI.federaciones.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb0:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb0

  couchdb1:
    container_name: couchdb1
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "6984:5984"
    networks:
      - default

  RSCE.federaciones.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb1:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb1

  couchdb2:
    container_name: couchdb2
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "7984:5984"
    networks:
      - default

  TKC.federaciones.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb2:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb2
```



```
couchdb3:
  container_name: couchdb3
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "8984:5984"
  networks:
    - default

ACW.federaciones.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb3:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb3
```

Y en el segundo servidor únicamente se dejará lo relativo a los peers de la organización ColegiosVeterinarios:

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

networks:
  default:
    external:
      name: netcan

services:
  couchdb4:
    container_name: couchdb4
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "9984:5984"
    networks:
      - default

  CVMadrid.colegiosveterinarios.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb4:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb4

  couchdb5:
    container_name: couchdb5
    image: hyperledger/fabric-couchdb:0.4.15
    environment:
      - COUCHDB_USER=
      - COUCHDB_PASSWORD=
    ports:
      - "10984:5984"
    networks:
      - default

  CVAndalucia.colegiosveterinarios.netcan.com:
    environment:
      - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
      - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb5:5984
      - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
      - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
    depends_on:
      - couchdb5
```



```
couchdb6:
  container_name: couchdb6
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "11984:5984"
  networks:
    - default

CVAragon.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb6:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb6

couchdb7:
  container_name: couchdb7
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "12984:5984"
  networks:
    - default

CVAsturias.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb7:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb7

couchdb8:
  container_name: couchdb8
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "13984:5984"
  networks:
    - default

CVIlesBalears.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb8:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb8

couchdb9:
  container_name: couchdb9
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "14984:5984"
  networks:
    - default

CVCanarias.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb9:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb9
```



```
couchdb10:
  container_name: couchdb10
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "15984:5984"
  networks:
    - default

CVCantabria.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb10:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb10

couchdb11:
  container_name: couchdb11
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "16984:5984"
  networks:
    - default

CVCastillayLeon.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb11:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb11

couchdb12:
  container_name: couchdb12
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "17984:5984"
  networks:
    - default

CVCastillaLaMancha.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb12:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb12

couchdb13:
  container_name: couchdb13
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "18984:5984"
  networks:
    - default

CVCataluna.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb13:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb13
```



```
couchdb14:
  container_name: couchdb14
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "19984:5984"
  networks:
    - default

CVComunitatValenciana.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb14:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb14

couchdb15:
  container_name: couchdb15
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "20984:5984"
  networks:
    - default

CVExtremadura.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb15:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb15

couchdb16:
  container_name: couchdb16
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "21984:5984"
  networks:
    - default

CVGalicia.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb16:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb16

couchdb17:
  container_name: couchdb17
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "22984:5984"
  networks:
    - default

CVMurcia.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb17:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb17
```



```
couchdb18:
  container_name: couchdb18
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "23984:5984"
  networks:
    - default

CVNavarra.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb18:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb18

couchdb19:
  container_name: couchdb19
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "24984:5984"
  networks:
    - default

CVPaisVasco.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb19:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb19

couchdb20:
  container_name: couchdb20
  image: hyperledger/fabric-couchdb:0.4.15
  environment:
    - COUCHDB_USER=
    - COUCHDB_PASSWORD=
  ports:
    - "25984:5984"
  networks:
    - default

CVLaRioja.colegiosveterinarios.netcan.com:
  environment:
    - CORE_LEDGER_STATE_STATEDATABASE=CouchDB
    - CORE_LEDGER_STATE_COUCHDBCONFIG_COUCHDBADDRESS=couchdb20:5984
    - CORE_LEDGER_STATE_COUCHDBCONFIG_USERNAME=
    - CORE_LEDGER_STATE_COUCHDBCONFIG_PASSWORD=
  depends_on:
    - couchdb20
```

En el archivo docker-compose-base.yaml, en el primer servidor se deja lo relativo al orderer y los peers de la organización Federaciones:

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

services:

  orderer.netcan.com:
    container_name: orderer.netcan.com
    image: hyperledger/fabric-orderer:1.4.0
    environment:
      - ORDERER_GENERAL_LOGLEVEL=INFO
```



```
- ORDERER_GENERAL_LISTENADDRESS=0.0.0.0
- ORDERER_GENERAL_GENESISMETHOD=file
- ORDERER_GENERAL_GENESISFILE=/var/hyperledger/orderer/orderer.genesis.block
- ORDERER_GENERAL_LOCALMSPID=OrdererMSP
- ORDERER_GENERAL_LOCALMSPDIR=/var/hyperledger/orderer/msp
# enabled TLS
- ORDERER_GENERAL_TLS_ENABLED=true
- ORDERER_GENERAL_TLS_PRIVATEKEY=/var/hyperledger/orderer/tls/server.key
- ORDERER_GENERAL_TLS_CERTIFICATE=/var/hyperledger/orderer/tls/server.crt
- ORDERER_GENERAL_TLS_ROOTCAS=[/var/hyperledger/orderer/tls/ca.crt]
- CORE_VM_DOCKER_HOSTCONFIG_NETWORKMODE=tfm_netcan
working_dir: /opt/gopath/src/github.com/hyperledger/fabric
command: orderer
volumes:
- ./channel-artifacts/genesis.block:/var/hyperledger/orderer/orderer.genesis.block
- ./crypto-config/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp:/var/hyperledger/orderer/msp
- ./crypto-config/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/tls:/var/hyperledger/orderer/tls
- orderer.netcan.com:/var/hyperledger/production/orderer
ports:
- 7050:7050

FCI.federaciones.netcan.com:
container_name: FCI.federaciones.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=FCI.federaciones.netcan.com
- CORE_PEER_ADDRESS=FCI.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=RSCE.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=FCI.federaciones.netcan.com:7051
- CORE_PEER_LOCALMSPID=FederacionesMSP
volumes:
- /var/run/:/host/var/run/
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/FCI.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/FCI.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
- FCI.federaciones.netcan.com:/var/hyperledger/production
ports:
- 7051:7051
- 7053:7053

RSCE.federaciones.netcan.com:
container_name: RSCE.federaciones.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=RSCE.federaciones.netcan.com
- CORE_PEER_ADDRESS=RSCE.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=RSCE.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=FCI.federaciones.netcan.com:7051
- CORE_PEER_LOCALMSPID=FederacionesMSP
volumes:
- /var/run/:/host/var/run/
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/RSCE.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/RSCE.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
- RSCE.federaciones.netcan.com:/var/hyperledger/production
ports:
- 8051:7051
- 8053:7053

TKC.federaciones.netcan.com:
container_name: TKC.federaciones.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=TKC.federaciones.netcan.com
- CORE_PEER_ADDRESS=TKC.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=TKC.federaciones.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=FCI.federaciones.netcan.com:7051
- CORE_PEER_LOCALMSPID=FederacionesMSP
volumes:
- /var/run/:/host/var/run/
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/TKC.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
```



```
- ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/TKC.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
  - TKC.federaciones.netcan.com:/var/hyperledger/production
ports:
  - 9051:7051
  - 9053:7053

ACW.federaciones.netcan.com:
  container_name: ACW.federaciones.netcan.com
  extends:
    file: peer-base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=ACW.federaciones.netcan.com
    - CORE_PEER_ADDRESS=ACW.federaciones.netcan.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=ACW.federaciones.netcan.com:7051
    - CORE_PEER_GOSSIP_BOOTSTRAP=FCI.federaciones.netcan.com:7051
    - CORE_PEER_LOCALMSPID=FederacionesMSP
  volumes:
    - /var/run/:/host/var/run/
    - ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/ACW.federaciones.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ./crypto-
config/peerOrganizations/federaciones.netcan.com/peers/ACW.federaciones.netcan.com/tls:/etc/hyperledger/fabric/tls
  - ACW.federaciones.netcan.com:/var/hyperledger/production
ports:
  - 10051:7051
  - 10053:7053
```

Y en el segundo servidor se deja lo relativo a los peers de la organización ColegiosVeterinarios:

```
# Copyright IBM Corp. All Rights Reserved.
#
# SPDX-License-Identifier: Apache-2.0
#
version: '2'

services:

  CVMadrid.collegiosveterinarios.netcan.com:
    container_name: CVMadrid.collegiosveterinario.netcan.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=CVMadrid.collegiosveterinarios.netcan.com
      - CORE_PEER_ADDRESS=CVMadrid.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVMadrid.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=CVAndalucia.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
    volumes:
      - /var/run/:/host/var/run/
      - ./crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVMadrid.collegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ./crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVMadrid.collegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
  - CVMadrid.collegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
  - 11051:7051
  - 11053:7053

  CVAndalucia.collegiosveterinarios.netcan.com:
    container_name: CVAndalucia.collegiosveterinarios.netcan.com
    extends:
      file: peer-base.yaml
      service: peer-base
    environment:
      - CORE_PEER_ID=CVAndalucia.collegiosveterinarios.netcan.com
      - CORE_PEER_ADDRESS=CVAndalucia.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVAndalucia.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.collegiosveterinarios.netcan.com:7051
      - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
```



```
volumes:
  - /var/run/:/host/var/run/
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAndalucia.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAndalucia.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
    - CVAndalucia.colegiosveterinarios.netcan.com:/var/hyperledger/production
  ports:
    - 12051:7051
    - 12053:7053

CVAragon.colegiosveterinarios.netcan.com:
  container_name: CVAragon.colegiosveterinarios.netcan.com
  extends:
    file: peer-base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=CVAragon.colegiosveterinarios.netcan.com
    - CORE_PEER_ADDRESS=CVAragon.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVAragon.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_BOOTSTRAP=CMadrid.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
  volumes:
    - /var/run/:/host/var/run/
    - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAragon.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAragon.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
    - CVAragon.colegiosveterinarios.netcan.com:/var/hyperledger/production
  ports:
    - 13051:7051
    - 13053:7053

CVAsturias.colegiosveterinarios.netcan.com:
  container_name: CVAsturias.colegiosveterinarios.netcan.com
  extends:
    file: peer-base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=CVAsturias.colegiosveterinarios.netcan.com
    - CORE_PEER_ADDRESS=CVAsturias.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVAsturias.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_BOOTSTRAP=CMadrid.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
  volumes:
    - /var/run/:/host/var/run/
    - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAsturias.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVAsturias.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
    - CVAsturias.colegiosveterinarios.netcan.com:/var/hyperledger/production
  ports:
    - 14051:7051
    - 14053:7053

CVIllesBalears.colegiosveterinarios.netcan.com:
  container_name: CVIllesBalears.colegiosveterinarios.netcan.com
  extends:
    file: peer-base.yaml
    service: peer-base
  environment:
    - CORE_PEER_ID=CVIllesBalears.colegiosveterinarios.netcan.com
    - CORE_PEER_ADDRESS=CVIllesBalears.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVIllesBalears.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_GOSSIP_BOOTSTRAP=CMadrid.colegiosveterinarios.netcan.com:7051
    - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
  volumes:
    - /var/run/:/host/var/run/
    - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVIllesBalears.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVIllesBalears.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
```



```
- CVIllesBalears.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 15051:7051
- 15053:7053

CVCanarias.colegiosveterinarios.netcan.com:
container_name: CVCanarias.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=CVCanarias.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVCanarias.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVCanarias.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCanarias.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCanarias.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVCanarias.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 16051:7051
- 16053:7053

CVCantabria.colegiosveterinarios.netcan.com:
container_name: CVCantabria.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=CVCantabria.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVCantabria.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVCantabria.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCantabria.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCantabria.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVCantabria.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 17051:7051
- 17053:7053

CVCastillayLeon.colegiosveterinarios.netcan.com:
container_name: CVCastillayLeon.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=CVCastillayLeon.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVCastillayLeon.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVCastillayLeon.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCastillayLeon.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCastillayLeon.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVCastillayLeon.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 18051:7051
- 18053:7053

CVCastillalaMancha.colegiosveterinarios.netcan.com:
container_name: CVCastillalaMancha.colegiosveterinarios.netcan.com
extends:
```



```
file: peer-base.yaml
service: peer-base
environment:
- CORE_PEER_ID=CVCastillalaMancha.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVCastillalaMancha.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVCastillalaMancha.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCastillalaMancha.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCastillalaMancha.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVCastillalaMancha.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 19051:7051
- 19053:7053

CVCataluna.colegiosveterinarios.netcan.com:
container_name: CVCataluna.colegiosveterinarios.netcan.com
extends:
file: peer-base.yaml
service: peer-base
environment:
- CORE_PEER_ID=CVCataluna.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVCataluna.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVCataluna.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCataluna.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVCataluna.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVCCataluna.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 20051:7051
- 20053:7053

CVComunitatValenciana.colegiosveterinarios.netcan.com:
container_name: CVComunitatValenciana.colegiosveterinarios.netcan.com
extends:
file: peer-base.yaml
service: peer-base
environment:
- CORE_PEER_ID=CVComunitatValenciana.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVComunitatValenciana.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVComunitatValenciana.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVComunitatValenciana.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVComunitatValenciana.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVComunitatValenciana.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 21051:7051
- 21053:7053

CVExtremadura.colegiosveterinarios.netcan.com:
container_name: CVExtremadura.colegiosveterinarios.netcan.com
extends:
file: peer-base.yaml
service: peer-base
environment:
- CORE_PEER_ID=CVExtremadura.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVExtremadura.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVExtremadura.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
```



```
volumes:
  - /var/run/:/host/var/run/
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVExtremadura.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVExtremadura.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
  - CVExtremadura.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
  - 22051:7051
  - 22053:7053

CVGalicia.colegiosveterinarios.netcan.com:
  container_name: CVGalicia.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
  - CORE_PEER_ID=CVGalicia.colegiosveterinarios.netcan.com
  - CORE_PEER_ADDRESS=CVGalicia.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVGalicia.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
  - /var/run/:/host/var/run/
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVGalicia.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVGalicia.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
  - CVGalicia.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
  - 23051:7051
  - 23053:7053

CVMurcia.colegiosveterinarios.netcan.com:
  container_name: CVMurcia.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
  - CORE_PEER_ID=CVMurcia.colegiosveterinarios.netcan.com
  - CORE_PEER_ADDRESS=CVMurcia.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVMurcia.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
  - /var/run/:/host/var/run/
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVMurcia.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVMurcia.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
  - CVMurcia.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
  - 24051:7051
  - 24053:7053

CVNavarra.colegiosveterinarios.netcan.com:
  container_name: CVNavarra.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
  - CORE_PEER_ID=CVNavarra.colegiosveterinarios.netcan.com
  - CORE_PEER_ADDRESS=CVNavarra.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVNavarra.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
  - CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
  - /var/run/:/host/var/run/
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVNavarra.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
  - ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVNavarra.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
```



```
- CVNavarra.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 25051:7051
- 25053:7053

CVPaisVasco.colegiosveterinarios.netcan.com:
container_name: CVPaisVasco.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=CVPaisVasco.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVPaisVasco.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVPaisVasco.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVPaisVasco.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVPaisVasco.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVPaisVasco.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 26051:7051
- 26053:7053

CVLaRioja.colegiosveterinarios.netcan.com:
container_name: CVLaRioja.colegiosveterinarios.netcan.com
extends:
  file: peer-base.yaml
  service: peer-base
environment:
- CORE_PEER_ID=CVLaRioja.colegiosveterinarios.netcan.com
- CORE_PEER_ADDRESS=CVLaRioja.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_EXTERNALENDPOINT=CVLaRioja.colegiosveterinarios.netcan.com:7051
- CORE_PEER_GOSSIP_BOOTSTRAP=CVMadrid.colegiosveterinarios.netcan.com:7051
- CORE_PEER_LOCALMSPID=ColegiosVeterinariosMSP
volumes:
- /var/run/:/host/var/run/
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVLaRioja.colegiosveterinarios.netcan.com/msp:/etc/hyperledger/fabric/msp
- ../crypto-
config/peerOrganizations/colegiosveterinarios.netcan.com/peers/CVLaRioja.colegiosveterinarios.netcan.com/tls:/etc/hyperledger/fabric/tls
- CVLaRioja.colegiosveterinarios.netcan.com:/var/hyperledger/production
ports:
- 27051:7051
- 27053:7053
```

El archivo peer-base.yaml será idéntico para los dos servidores y no varía con respecto a la red desplegada en un servidor, al igual que el resto de archivos que hay que copiar al segundo servidor: material criptográfico y artefactos creados.

En cuanto a los scripts la única diferencia con la red anterior es que se copiarán desde la carpeta scripts3 del Github y que se utilizará el script Serv2_script.sh que en las redes anteriores no se había utilizado.

Al lanzar el script de inicio (netcan_script.sh) la red se levantará de la misma manera aunque primero se levantarán la red Docker Swarm, luego lo harán los peers de la organización Federaciones:



Y posteriormente los de la organización ColegiosVeterinarios en el segundo servidor, continuando después el despliegue de la red exactamente igual que en la red anterior.

```
Creating volume "tfm_CVAndalucia.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVAragon.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVAsturias.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVIllesBalears.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVCanarias.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVCantabria.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVCastillaLeon.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVCastillaLaMancha.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVCataluna.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVComunitatValenciana.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVExtramadura.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVGalicia.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVMadrid.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVMurcia.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVNavarra.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVPaisVasco.colegiosveterinarios.netcan.com" with default driver
Creating volume "tfm_CVLARioja.colegiosveterinarios.netcan.com" with default driver
Creating couchdb8 ... done
Creating couchdb17 ... done
Creating couchdb5 ... done
Creating couchdb4 ... done
Creating couchdb14 ... done
Creating couchdb16 ... done
Creating couchdb20 ... done
Creating couchdb11 ... done
Creating couchdb7 ... done
```

Finalizando con la carga inicial de datos.

```
*****
Cargando datos de VACUNACIONES
*****
2020-09-12 23:35:36.518 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:2
00 payload:"{\\"docType\":\"\\\"CONTADOR\\\",\\\"IDMaximo\\\":11}"
2020-09-12 23:35:36.580 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:2
00 payload:"89"
2020-09-12 23:35:43.768 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:2
00 payload:"{\\"docType\":\"\\\"CONTADOR\\\",\\\"IDMaximo\\\":89}"
2020-09-12 23:35:44.090 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:2
00 payload:"514"
2020-09-12 23:35:51.190 UTC [chaincodeCmd] chaincodeInvokeOrQuery -> INFO 001 Chaincode invoke successful. result: status:2
00 payload:"{\\"docType\":\"\\\"CONTADOR\\\",\\\"IDMaximo\\\":512}"
*****
DATOS INICIALES CARGADOS
*****
```

Para apagar la red utilizaremos el script `stop_netcan_script.sh` que limpia los dockers y los chaincodes desplegados.



Bibliografía

- HYPERLEDGER.ORG, Hyperledger Fabric release 1.4, 2017, <https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html>
- HYPERLEDGER.ORG, Hyperledger Fabric docs release 1.4, 2017, <https://github.com/hyperledger/fabric/tree/release-1.4/docs>
- TORRES PALOMINO, SERGIO, Workshop Hyperledger Fabric – Despliegue de una red Fabric para Universidades, 1 de octubre de 2018, <https://blocknitive.com/blog/workshop-hyperledger-fabric-despliegue-de-una-red-fabric-para-universidades/>
- FABFILE.ORG, Welcome to Fabric's documentation, 2017, <https://docs.fabfile.org/en/2.5/>
- HYPERLEDGER.ORG, Using CouchDB, 2020, https://hyperledger-fabric.readthedocs.io/en/release-2.2/couchdb_tutorial.html
- HYPERLEDGER.ORG, CouchDB as the State Database, 2020, https://hyperledger-fabric.readthedocs.io/en/release-2.2/couchdb_as_state_database.html#couchdb-configuration
- J. CHRIS ANDERSON, JAN LEHNARDT AND NOAH SLATER, CouchDB The Definitive Guide, 2018, <http://guide.couchdb.org/draft/index.html#>
- HYPERLEDGER.ORG, Fabric CA User's Guide, 2017, <https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/users-guide.html>
- HYPERLEDGER.ORG, Fabric CA Operations Guide, 2017, https://hyperledger-fabric-ca.readthedocs.io/en/release-1.4/operations_guide.html
- HYPERLEDGER.ORG, Blockchain-explorer, 2018, <https://github.com/hyperledger/blockchain-explorer/tree/release-3.9>
- HYPERLEDGER.ORG, Hyperledger Explorer DocumentationExplorer, 14 de julio de 2020, <https://readthedocs.org/projects/blockchain-explorer/downloads/pdf/master/>
- HYPERLEDGER.ORG, Tutorials, 2020, <https://hyperledger-fabric.readthedocs.io/en/latest/tutorials.html>
- HYPERLEDGER.ORG, Troubleshooting TechNotes - Hyperledger Explorer, <https://github.com/hyperledger/blockchain-explorer/blob/master/TROUBLESHOOT.md>
- HYPERLEDGER.ORG, Hyperledger Caliper, 2020, <https://www.hyperledger.org/use/caliper>
- PAVAN ADHAV, Caliper integration with Hyperledger fabric, 6 DE ENERO DE 2020, <https://medium.com/coinmonks/caliper-integration-with-hyperledger-fabric-d2aa0df2f73f>
- HYPERLEDGER.ORG, Hyperledger Caliper - Fabric, 2020, <https://hyperledger.github.io/caliper/v0.3.2/fabric-config/#network-configuration-file-reference>



VERSIÓN
FINAL

15-09-2020

Blockchain & BigData Canino

ANEXO III: Chaincodes y funcionalidades
del sistema



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE

Contratos divididos por Organizaciones	4
Federaciones Caninas	4
Colegios de Veterinarios.....	4
Contratos inteligentes	5
Parámetros de entrada/salida de las funciones.....	5
Funciones comunes de los contratos	6
ejecutarConsulta.....	6
asignarEstado	6
borrarEstado.....	6
consultarEstado.....	6
consultarRangoEstados	6
getQueryResultForQueryString.....	6
PERSONAS.....	7
registrarPersona	8
registrarDocumentoIdentidad	9
modificarNombreApellidos.....	9
Funciones comunes.....	10
AFIJOS.....	13
registrarAfijo.....	13
registrarCambioPropietario.....	15
registrarCancelacionAfijo.....	16
cargarDatosIniciales	18
cargarDatosIniciales_Propietarios	18
consultarDatosAfijo	19
obtenerCertificadoAfijo.....	19
Funciones comunes.....	19

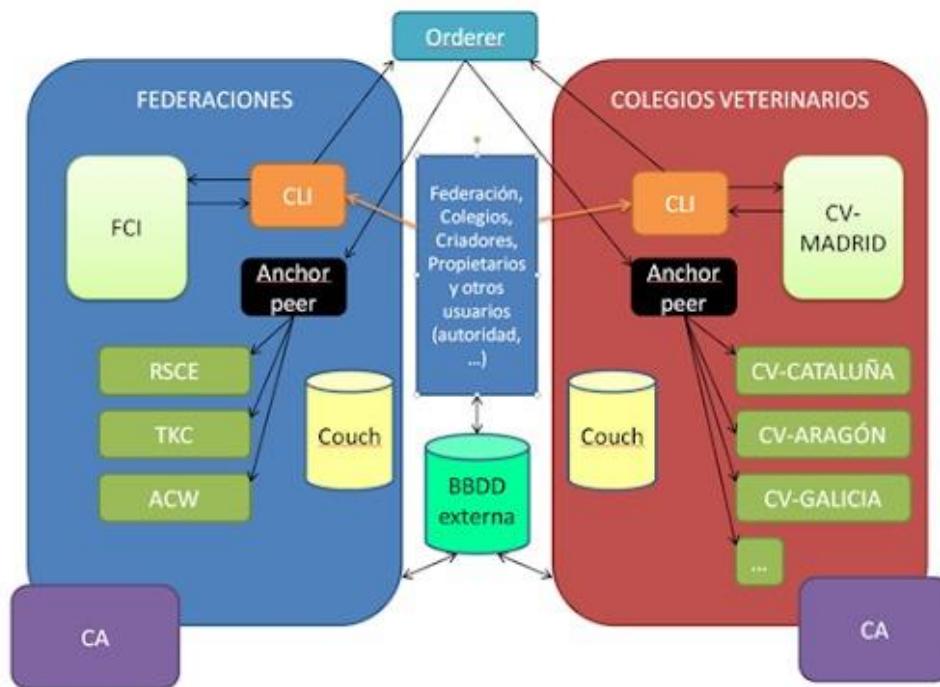


PERROS	22
registrarPerro	22
registrarCambioPropietario.....	25
registrarDefuncionPerro	27
registrarReconocimientoRaza	29
cargarDatosIniciales	30
cargarDatosIniciales_Propietarios	31
consultarDatosEjemplar	32
obtenerCertificadoRegistro.....	32
obtenerPedigri	32
registrarCesionTemporal	32
Funciones comunes.....	32
SOLICITUDES.....	37
solicitarRegistrarCamada.....	40
solicitarRegistrarPerro.....	40
solicitarRegistrarCambioPropietarioPerro	40
solicitarRegistrarCambioPropietarioAfijo.....	40
solicitarRegistrarCancelacionAfijo	40
validarSolicitud.....	41
cargarDatosIniciales	42
cargarDatosIniciales_Autorizaciones	43
Funciones complementarias.....	44
Funciones comunes.....	44
MICROCHIP	48
registrarMicrochips	48
consultarMicrochip	49
cargarDatosIniciales	49
Funciones comunes.....	50
VACUNAS	52
registrarVacuna	53
obtenerCertificadoVacunaciones (pasaporte)	54
consultarVacunaciones	54
cargarDatosIniciales	55



cargarDatosIniciales_VacunasPerrosProteccion	55
cargarDatosIniciales_VacunasProteccion	56
Funciones comunes	57
PERFILES	59
registrarPerfilPersona	59
cancelarPerfilPersona	60
cargarDatosIniciales	62
Funciones complementarias	62
Funciones comunes	63
RAZAS	65
Funciones comunes	66
VETERINARIOS	68
registrarColegiaturaPersona	68
cancelarPerfilPersona	69
cargarDatosIniciales	71
Funciones comunes	71
Otros posibles contratos inteligentes	73
TITULOS	73
solicitarTitulo	73
obtenerTitulo	73
consultarTitulos	73
EXPOSICIONES Y CONCURSOS	74
registarShow	74
registarResultadoShow	74
cagarResultadosShow	74
consultarResultadosShow	74
ADN	75
ENFERMEDADES / TRATAMIENTOS	75
Carga inicial de datos en la Blockchain	76
Introducción	76
Proceso de carga	76

Contratos divididos por Organizaciones



Los contratos que utiliza cada organización son los siguientes

Federaciones Caninas

- Personas
- Perfil
- Perros
- Afijos
- Solicitudes
- Veterinarios
- Microchip
- Razas
- Exposiciones y concursos
- Títulos

Colegios de Veterinarios

- Personas
- Perfil
- Perros
- Veterinarios
- Vacunas
- Microchips
- Razas

Contratos inteligentes

Parámetros de entrada/salida de las funciones

Las funciones definidas en los contratos disponen de parámetros de entrada específicos para su funcionamiento.

Como regla general dispones 2 argumentos:

```
args[0] : de tipo JSON con datos específicos de la función  
args[1]: de tipo JSON con los datos de seguridad donde se identifica al usuario que invoca la función.
```

Por el contrario, todas las funciones definidas en los contratos tendrán un parámetro de salida de tipo Response:

```
type Response struct {  
    // A status code that should follow the HTTP status codes.  
    Status int32 `protobuf:"varint,1,opt,name=status,proto3"  
    json:"status,omitempty"`  
    // A message associated with the response code.  
    Message string `protobuf:"bytes,2,opt,name=message,proto3"  
    json:"message,omitempty"`  
    // A payload that can be used to include metadata with this response.  
    Payload []byte `protobuf:"bytes,3,opt,name=payload,proto3"  
    json:"payload,omitempty"  
    XXX_NoUnkeyedLiteral struct{} `json:"-"  
    XXX_unrecognized []byte `json:"-"  
    XXX_sizecache int32 `json:"-"`
}
```





Funciones comunes de los contratos

Existen un conjunto de funciones que se definen dentro de los contratos:

ejecutarConsulta

Permite realizar consultas al sistema a través del lenguaje de consulta de Mango, que se expresa como un objeto JSON que describe los filtros de los documentos de interés que se desean consultar.

asignarEstado

Asigna un valor de estado al sistema.

borrarEstado

Elimina un valor de estado del sistema.

consultarEstado

Consulta un valor de estado del sistema.

consultarRangoEstados

Consulta los valores de estado del sistema comprendidos entre dos valores.

getQueryResultForQueryString

Devuelve en el siguiente formato JSON el resultado de una consulta, donde el TipoEstado tendrá la definición específica del estado consultado.

```
type TipoQueryEstado struct {
    Key      string           `json:"Key"`
    Record  TipoEstado `json:"Record"`
}
```



PERSONAS

Este contrato inteligente es el encargado de gestionar los datos de las personas que intervienen en las solicitudes de los usuarios del sistema.

Actualmente gestiona los datos identificativos de:

- Criadores (propietarios de afijos)
- Propietarios de Perros
- Veterinarios
- Miembros de Federaciones

El presente contrato inteligente esta diseñado para admitir en el futuro ampliaciones de la blockchain, dando la posibilidad de albergar los datos de identificación de otro tipo de usuarios o datos asociados.

Por ejemplo, incorporar:

- Jueces de certámenes caninos
- Miembros de Asociaciones caninas
- Miembros de Club de raza caninas y de grupos
- Miembros de laboratorios (para registros de ADN)
- ...





Funciones que incorpora el contrato inteligente:

registrarPersona

Introduce en el sistema los datos necesarios para la identificación de una nueva persona.

- **Argumentos entrada:**

```
type registrarPersonas struct {
    Nombre           string `json:"Nombre"`
    Apellido1       string `json:"Apellido1"`
    Apellido2       string `json:"Apellido2"`
    TipoDocumento   string `json:"TipoDocumento"`
    IdentificadorDocumento string `json:"IdentificadorDocumento"`
    PaisEmisor      string `json:"PaisEmisor"`
    Certificado     string `json:"Certificado"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 1
- El formato JSON de args[0] sea valido
- Los argumentos Nombre, Apellido1, Apellido2, TipoDocumento, IdentificadorDocumento y PaisEmisor tenga un valor.
- No exista otra persona registrada con esos mismo TipoDocumento, IdentificadorDocumento y PaisEmisor.

- **Argumentos salida:**

Devuelve el registro insertado:

```
type Personas struct {
    ObjectType        string `json:"docType"`
    IDPersona         int    `json:"IDPersona"`
    Nombre            string `json:"Nombre"`
    Apellido1         string `json:"Apellido1"`
    Apellido2         string `json:"Apellido2"`
    TipoDocumento    string `json:"TipoDocumento"`
    IdentificadorDocumento string `json:"IdentificadorDocumento"`
    PaisEmisor        string `json:"PaisEmisor"`
    Certificado      string `json:"Certificado"`
    FechaAlta         string `json:"FechaAlta"`
    FechaBaja         string `json:"FechaBaja"`
}
```



registrarDocumentoIdentidad

Introduce en el sistema los datos otros documentos de identidad de la persona, válidos para identificar a una persona registra.

Por ejemplo: las mujeres rumanas cuando se casa cambian el documento de identidad por el de su marido añadiendo un digito adicional o en España cuando un persona adquiere la nacionalidad se le cambia el NIE por el NIF.

modificarNombreApellidos

Modifica en el sistema el nombre y/o apellido de una persona registra.

Por ejemplo: en caso de cambio de nombre como Ignacio por Iñaki o cambio de orden de los apellidos.





cargarDatosIniciales

Registrar en el sistema los datos de las propietarios, veterinarios y personal de las federaciones definidos en un fichero de texto con el siguiente formato JSON:

```
type Personas struct {
    ObjectType      string `json:"docType"`
    IDPersona       int    `json:"IDPersona"`
    Nombre          string `json:"Nombre"`
    Apellido1       string `json:"Apellido1"`
    Apellido2       string `json:"Apellido2"`
    TipoDocumento   string `json:"TipoDocumento"`
    IdentificadorDocumento string `json:"IdentificadorDocumento"`
    PaisEmisor      string `json:"PaisEmisor"`
    Certificado     string `json:"Certificado"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

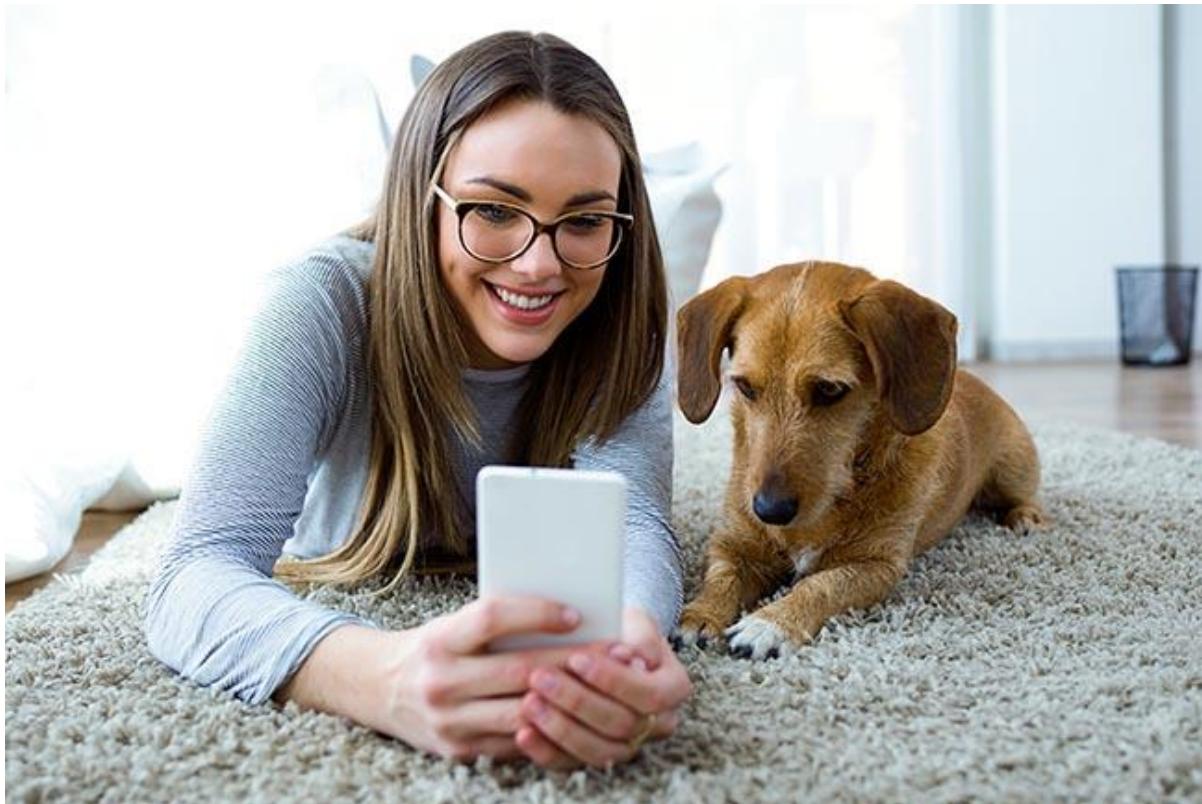
Devuelve el número de registros insertados:

```
NumeroRegistros string
```

Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**



Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# PERSONAS
peer chaincode invoke -n personas -c
'{"function":"registrarPersona","Args":[{"\"Nombre\":\"Unai\",\"Apellido1\":\"Ares\",
\"Apellido2\":\"Icaran\",\"TipoDocumento\":\"NIF\",\"IdentificadorDocumento\":\
"30624315E\", \"PaisEmisor\":\"Spain\"}]}' -o $ORDENER_URL --tls --cafile $CA_FILE
-C $CHANNEL_NAME

# ERRORES PERSONAS
```



```
peer chaincode invoke -n personas -c '{"function":"registrarPersona","Args":[]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n personas -c  
'{"function":"registrarPersona","Args":[{"Nombre\" : \"Unai\", \"Apellido1\" : \"Ares\" , \"Apellido2\" : \"Icaran\", \"TipoDocumento\" : \"NIF\", \"IdentificadorDocumento\" : \"30624315E\", \"PaisEmisor\" : \"Spain\""}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n personas -c  
'{"function":"registrarPersona","Args":[{"\"Nombre\" : \"Unai\", \"Apellido1\" : \"Ares\" , \"Apellido2\" : \"Icaran\", \"TipoDocumento\" : \"NIF\", \"IdentificadorDocumento\" : \"30624315E\", \"PaisEmisor\" : \"Spain\""}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n personas -c  
'{"function":"registrarPersona","Args":[{"\"Apellido1\" : \"Ares\", \"Apellido2\" : \"Icaran\", \"TipoDocumento\" : \"NIF\", \"IdentificadorDocumento\" : \"30624315E\", \"PaisEmisor\" : \"Spain\""}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

CARGAS INICIALES

```
peer chaincode invoke -n personas -c  
'{"function":"cargarDatosIniciales","Args":[ "./json/personas_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
sleep 5
```

```
peer chaincode invoke -n personas -c  
'{"function":"asignarEstado","Args":["PERSONAS",  
"{"docType\" : \"CONTADOR\", \"IDMaximo\" : 1000}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```





AFIJOS

Este contrato inteligente es el encargado de gestionar los datos de los afijos de los criadores del sistema.

Un **afijo de un criador** es una denominación o marca personal registrado a través de una sociedad o federación canina que le autoriza a su titular o titulares a utilizarlo en la inscripción de camadas.



Funciones que incorpora el contrato inteligente:

registrarAfijo

Introduce en el sistema los datos necesarios para registrar y autorizar el uso de un nuevo Afijo de Criador por parte de uno o más usuarios cumpliendo los siguientes requisitos:

- Ningún solicitante puede ser propietario o copropietario de otro afijo
- El nombre afijo no debe existir
- **Argumentos entrada:**

```
type tipoRegistrarAfijoPropietario struct {
    IDPersona int `json:"IDPersona"`
}

type tipoRegistrarAfijo struct {
    Nombre      string
                                         `json:"Nombre"`
}
```



```
    Propietarios []tipoRegistrarAfijoPropietario `json:"Propietarios"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- Los argumentos Nombre y Propietarios tenga un valor.
- No existe otro Afijo con el mismo nombre
- Los propietarios están registrados
- Alguno de los propietarios no tiene actualmente otro afijo asignado como propietario o copropietario
- La persona que invoca la acción esta registrada y dispone de los permisos para realizar dicha acción

• **Argumentos salida:**

Devuelve los registros insertados:

```
type Afijos struct {
    ObjectType string `json:"docType"`
    IDAfijo    int    `json:"IDAfijo"`
    Nombre     string `json:"Nombre"`
    FechaAlta  string `json:"FechaAlta"`
    FechaBaja  string `json:"FechaBaja"`
}

type AfijosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDAfijoPropietario int `json:"IDAfijoPropietario"`
    IDAfijo        int    `json:"IDAfijo"`
    IDPersona       int    `json:"IDPersona"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}
```



registrarCambioPropietario

Registrar en el sistema los datos de los nuevos propietarios de un afijo, dando de baja el registro de los actuales propietarios.

- **Argumentos entrada:**

```
type tiporegistrarCancelacionAfijo struct {  
    IDAfijo int `json:"IDAfijo"  
}
```

```
type TipoSeguridad struct {  
    IDPersona int `json:"IDPersona"  
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDAfijo tenga un valor y corresponda a un afijo activo.
- Los nuevos propietarios están registrados y no tiene ninguno de ellos otro afijo asignado como propietario o copropietario
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción





- **Argumentos salida:**

Devuelve los registros insertados:

```
type Afijos struct {
    ObjectType string `json:"docType"`
    IDAfijo    int    `json:"IDAfijo"`
    Nombre     string `json:"Nombre"`
    FechaAlta  string `json:"FechaAlta"`
    FechaBaja  string `json:"FechaBaja"`
}

type AfijosPropietariosBaja struct {
    ObjectType      string `json:"docType"`
    IDAfijoPropietario int `json:"IDAfijoPropietario"`
    IDAfijo        int    `json:"IDAfijo"`
    IDPersona       int    `json:"IDPersona"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}

type AfijosPropietariosAlta struct {
    ObjectType      string `json:"docType"`
    IDAfijoPropietario int `json:"IDAfijoPropietario"`
    IDAfijo        int    `json:"IDAfijo"`
    IDPersona       int    `json:"IDPersona"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}
```

registrarCancelacionAfijo

Registrar en el sistema la cancelación del Afijo de Criador, cuyo nombre no podrá ser utilizado por ninguna otra persona.

- **Argumentos entrada:**

```
type tiporegistrarCancelacionAfijo struct {
    IDAfijo int `json:"IDAfijo"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```



El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDAfijo tenga un valor y corresponda a un afijo activo.
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

• **Argumentos salida:**

Devuelve los registros insertados:

```
type Afijos struct {
    ObjectType string `json:"docType"`
    IDAfijo    int    `json:"IDAfijo"`
    Nombre     string `json:"Nombre"`
    FechaAlta  string `json:"FechaAlta"`
    FechaBaja  string `json:"FechaBaja"`
}
type AfijosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDAfijoPropietario int   `json:"IDAfijoPropietario"`
    IDAfijo        int    `json:"IDAfijo"`
    IDPersona       int    `json:"IDPersona"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}
```



cargarDatosIniciales

Insertar en el sistema los datos los distintos Afijos registrados por las federaciones en un fichero de texto con el siguiente formato JSON:

```
type Afijos struct {
    ObjectType string `json:"docType"`
    IDAfijo    int    `json:"IDAfijo"`
    Nombre     string `json:"Nombre"`
    FechaAlta  string `json:"FechaAlta"`
    FechaBaja  string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

cargarDatosIniciales_Propietarios

Insertar en el sistema los datos los Propietarios de los distintos Afijos registrados por las federaciones en un fichero de texto con el siguiente formato JSON:

```
type AfijosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDAfijoPropietario int `json:"IDAfijoPropietario"`
    IDAfijo        int `json:"IDAfijo"`
    IDPersona       int `json:"IDPersona"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```



consultarDatosAfijo

Devuelve al solicitante los datos que sean públicos del afijo consultado.

obtenerCertificadoAfijo

Proporciona al propietario un fichero que certifica la propiedad de este.

Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**





Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# AFIJOS
peer chaincode invoke -n afijos -c
'{"function":"registrarAfijo","Args":[{"\"Nombre\":\"NUEVO
AFIJO\",\"Propietarios\":[{\\"IDPersona\":700},{\"IDPersona\":701},{\"IDPersona\":702}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCambioPropietario","Args": [ {"\"IDAfijo\":51,\"Propietarios\"
:[{\\"IDPersona\":800},{\"IDPersona\":801},{\"IDPersona\":802},{\"IDPersona\":803}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCancelacionAfijo","Args": [ {"\"IDAfijo\":51},\"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# ERRORES AFIJOS
peer chaincode invoke -n afijos -c
'{"function":"registrarAfijo","Args": [{"\"Nombre\":\"NUEVO
AFIJO\",\"Propietarios\":[{\\"IDPersona\":700},{\"IDPersona\":701},{\"IDPersona\":702}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarAfijo","Args": [{"\"Nombre\":\"OTRO
AFIJO\",\"Propietarios\":[{\\"IDPersona\":700},{\"IDPersona\":801},{\"IDPersona\":802}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCambioPropietario","Args": [ {"\"IDAfijo\":51,\"Propietarios\"
:[{\\"IDPersona\":1},{\"IDPersona\":2},{\"IDPersona\":3},{\"IDPersona\":4}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCambioPropietario","Args": [ {"\"IDAfijo\":1051,\"Propietario
s\":[{\\"IDPersona\":800},{\"IDPersona\":801},{\"IDPersona\":802},{\"IDPersona\":803}]}], \"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```



```
peer chaincode invoke -n afijos -c
'{"function":"registrarCambioPropietario","Args":[{"\"IDAfijo\":51,\"Propietarios\":[{\\"IDPersona\":8888888888888888},{\"IDPersona\":2},{\"IDPersona\":3},{\"IDPersona\":4}]},"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCancelacionAfijo","Args":[{"\"IDAfijo\":51}, {"\"IDPersona\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n afijos -c
'{"function":"registrarCambioPropietario","Args":[{"\"IDAfijo\":555555551,\"Propietarios\":[{\\"IDPersona\":800},{\"IDPersona\":801},{\"IDPersona\":802},{\"IDPersona\":803}]},"{\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

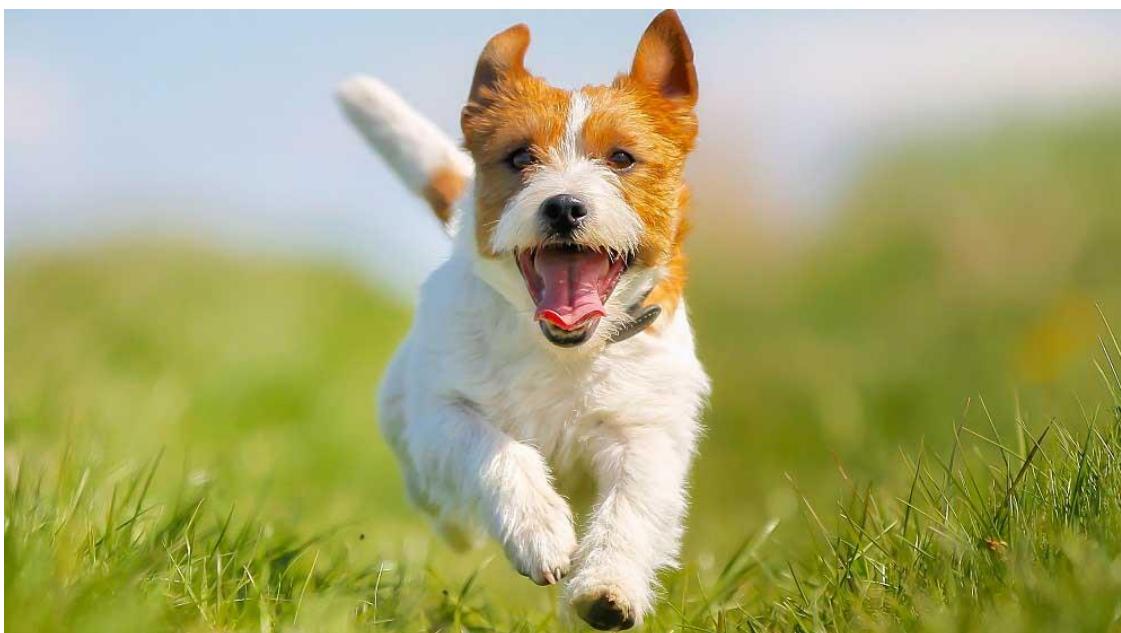
# CARGAS INICIALES

peer chaincode invoke -n $CC_NOMBRE_AFIJOS -c
'{"function":"cargarDatosIniciales","Args":["./json/afijos.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_AFIJOS -c
'{"function":"asignarEstado","Args":["AFIJOS",
"\"docType\":\"CONTADOR\", \"IDMaximo\":50}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_AFIJOS -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/afijos_propietarios.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_AFIJOS -c
'{"function":"asignarEstado","Args":["AFIJOS_PROPIETARIOS",
"\"docType\":\"CONTADOR\", \"IDMaximo\":50}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME
```





PERROS

Este contrato inteligente es el encargado de gestionar el registro de ejemplares canino, identificando su origen, pureza de raza, mencionando sus ascendentes y descendientes.



Permite registrar a los propietarios de ejemplar hembra registrar el nacimiento de una nueva camada en un periodo no superior a 1 mes desde su nacimiento (identificando a los progenitores y el nombre y sexo de cada cachorro)

También permite el registro de propiedad de perros mestizos en los cuales algún de los progenitores es desconocido o es un mestizo, y los cambios de propietarios, así como la defunción de los ejemplares.

Funciones que incorpora el contrato inteligente:

registrarPerro

Introduce en el sistema los datos necesarios para identificar el origen genealógico de uno o varios perros y registrar sus propietarios.



Permite registrar una camada o un ejemplar, determinando la pureza de raza y el afijo de criador según los criterios de validación detallados más adelante.

- **Argumentos entrada:**

```
type tipoRegistrarCamadaPerro struct {
    Nombre string `json:"Nombre"`
    IDSexo int     `json:"IDSexo"`
}

type tipoRegistrarCamadaPropietario struct {
    IDPersona int `json:"IDPersona"`
}

type tipoRegistrarCamada struct {
    Perros          []tipoRegistrarCamadaPerro      `json:"Perros"`
    IDPerroMadre   int                            `json:"IDPerroMadre"`
    IDPerroPadre   int                            `json:"IDPerroPadre"`
    IDAfijo        int                            `json:"IDAfijo"`
    IDRaza         int                            `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    Propietarios   []tipoRegistrarCamadaPropietario `json:"Propietarios"`
}
```

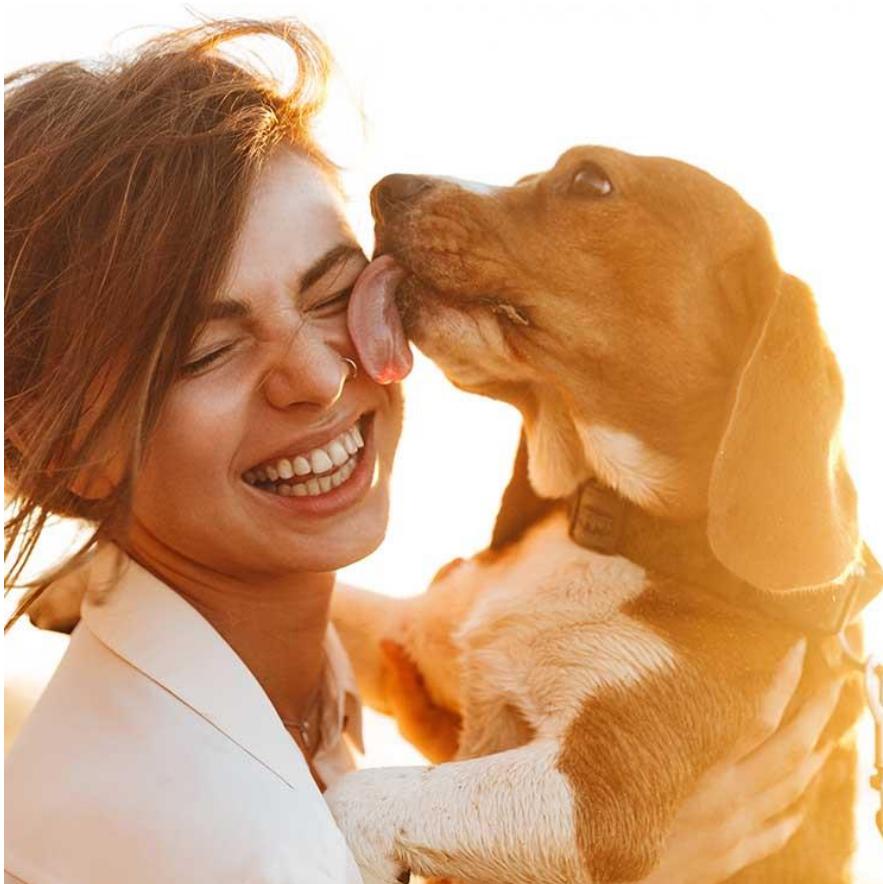
```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- En caso de que la madre no sea desconocida (IDMadre≠0):
 - El IDMadre está registrado y sigue viva.
 - El IDMadre este registrada como ejemplar Hembra (IDSexo = 0)
 - Su edad esta comprendida entre 1 y 10 años (sino perderá la pureza de raza y se asignará IDRaza = 0)
 - Si todos los propietarios de la Madre son propietarios del mismo Afijo de Criador (sino perderá la denominación del afijo, y se asignara IDAfijo = 0) que serán los propietarios del ejemplar
- En caso de que la madre sea desconocida (IDMadre = 0):
 - Los nuevos propietarios están registrados



- En caso de que el padre no sea desconocida ($IDPadre \neq 0$):
 - El IDPadre está registrado y sigue vivo.
 - El IDPadre este registrado como ejemplar Macho ($IDSexo = 1$)
 - Su edad está comprendida entre 1 y 12 años (sino perderá la pureza de raza y se asignará IDRaza = 0)
- Si la raza de los dos progenitores es igual (sino perderá la pureza de raza y se asignará IDRaza=0)
- Los argumentos Nombre e IDSexo tenga un valor valido:
 - En el caso de que IDAfijo $\neq 0$ no exista para ese afijo de criador otro ejemplar con el mismo nombre
 - IDSexo contiene o 0 (HEMBRA) o 1 (MACHO)
- El argumento FechaNacimiento tenga un valor q los últimos 30 días (sino perderá la pureza de raza y se asignará IDRaza = 0)
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción





- **Argumentos salida:**

Devuelve los registros insertados:

```
type Perros struct {
    ObjectType      string `json:"docType"`
    IDPerro         int    `json:"IDPerro"`
    Nombre          string `json:"Nombre"`
    IDAfijo         int    `json:"IDAfijo"`
    IDSexo          int    `json:"IDSexo"`
    IDPerroMadre   int    `json:"IDPerroMadre"`
    IDPerroPadre   int    `json:"IDPerroPadre"`
    IDRaza          int    `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    FechaDefuncion string `json:"FechaDefuncion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}

type PerrosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int `json:"IDPerroPropietario"`
    IDPerro         int    `json:"IDPerro"`
    IDPersona        int    `json:"IDPersona"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

registrarCambioPropietario

Registrar en el sistema los datos de los nuevos propietarios de un ejemplar, dando de baja el registro de los actuales propietarios.

- **Argumentos entrada:**

```
type TipoRegistrarCambioPropietarioPropietario struct {
    IDPersona int `json:"IDPersona"`
}

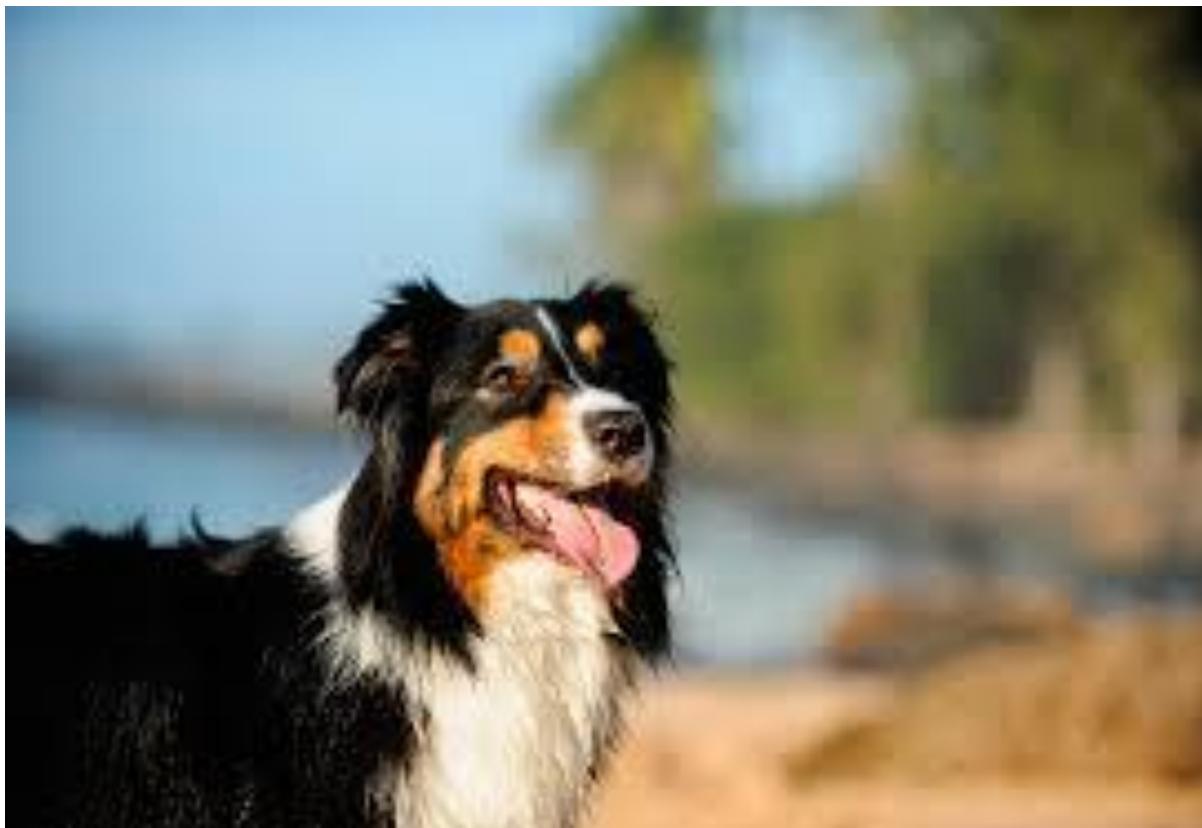
type TipoRegistrarCambioPropietario struct {
    IDPerro     int `json:"IDPerro"`
    Propietarios []TipoRegistrarCambioPropietarioPropietario `json:"Propietarios"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```



El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDAfijo tenga un valor y corresponda a un afijo activo.
- Los nuevos propietarios están registrados
- Alguno de los nuevos propietarios no tiene actualmente otro afijo asignado como propietario o copropietario
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción



• **Argumentos salida:**

Devuelve los registros insertados:

```
type Perros struct {
    ObjectType      string `json:"docType"`
    IDPerro        int    `json:"IDPerro"`
    Nombre         string `json:"Nombre"`
```



```
    IDAfijo      int   `json:"IDAfijo"`
    IDSexo       int   `json:"IDSexo"`
    IDPerroMadre int   `json:"IDPerroMadre"`
    IDPerroPadre int   `json:"IDPerroPadre"`
    IDRaza        int   `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    FechaDefuncion string `json:"FechaDefuncion"`
    FechaAlta     string `json:"FechaAlta"`
    FechaBaja     string `json:"FechaBaja"`
}

type PerrosPropietariosBaja struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int   `json:"IDPerroPropietario"`
    IDPerro          int   `json:"IDPerro"`
    IDPersona        int   `json:"IDPersona"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}

type PerrosPropietariosAlta struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int   `json:"IDPerroPropietario"`
    IDPerro          int   `json:"IDPerro"`
    IDPersona        int   `json:"IDPersona"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```

registrarDefuncionPerro

Registrar en el sistema la baja de un ejemplar por defunción.

- **Argumentos entrada:**

```
type tiporegistrarCancelacionAfijo struct {
    IDAfijo int `json:"IDAfijo"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPerro tenga un valor y corresponda a un ejemplar vivo.

- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

- **Argumentos salida:**

Devuelve los registros insertados:

```
type Perros struct {
    ObjectType      string `json:"docType"`
    IDPerro         int    `json:"IDPerro"`
    Nombre          string `json:"Nombre"`
    IDAfijo         int    `json:"IDAfijo"`
    IDSexo          int    `json:"IDSexo"`
    IDPerroMadre   int    `json:"IDPerroMadre"`
    IDPerroPadre   int    `json:"IDPerroPadre"`
    IDRaza          int    `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    FechaDefuncion string `json:"FechaDefuncion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}

type PerrosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int `json:"IDPerroPropietario"`
    IDPerro         int    `json:"IDPerro"`
    IDPersona        int    `json:"IDPersona"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```





registrarReconocimientoRaza

Registrar en el sistema la certificación de la Pueza de raza de un ejemplar cuyos ascendentes no están determinados o registrados en un libro de origen genealógico.

Este hecho se realiza de manera excepcional por una de las siguientes causas:

- Un juez especialista de Raza certifica en un evento organizado por las federaciones que el ejemplar cumple con todas las características del estándar de la raza.
- El propietario presenta una Certificado de un Libro de Origen Genealógico reconocido y la federación verifica la autenticación del documento con el organismo que gestiona el libro.

- **Argumentos entrada:**

```
type PerroReconocimientoRaza struct {
    IDPerro          int     `json:"IDPerro"`
    IDRaza           int     `json:"IDRaza"`
    Justificacion   string  `json:"Justificacion"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPerro tenga un valor y corresponda a un ejemplar vivo.
- El IDRaza tenga un valor y corresponda a un afijo activo.
- La Justificación tenga un valor
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

- **Argumentos salida:**

Devuelve los registros insertados:

```
type Perros struct {
    ObjectType      string `json:"docType"`
    IDPerro         int    `json:"IDPerro"`
    Nombre          string `json:"Nombre"`
    IDAfijo         int    `json:"IDAfijo"`
    IDSexo          int    `json:"IDSexo"`
    IDPerroMadre   int    `json:"IDPerroMadre"`
    IDPerroPadre   int    `json:"IDPerroPadre"`
    IDRaza          int    `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    FechaDefuncion string `json:"FechaDefuncion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}

type PerrosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int `json:"IDPerroPropietario"`
    IDPerro         int    `json:"IDPerro"`
    IDPersona        int    `json:"IDPersona"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```



cargarDatosIniciales

Registrar en el sistema los ejemplares definidos en un fichero de texto con el siguiente formato JSON:

```
type Perros struct {
    ObjectType      string `json:"docType"`
    IDPerro         int    `json:"IDPerro"`

    Nombre          string `json:"Nombre"`
    IDAfijo         int    `json:"IDAfijo"`
    IDSexo          int    `json:"IDSexo"`
    IDPerroMadre   int    `json:"IDPerroMadre"`
    IDPerroPadre   int    `json:"IDPerroPadre"`
    IDRaza          int    `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    FechaDefuncion string `json:"FechaDefuncion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```



```
Nombre      string `json:"Nombre"`
IDAfijo    int    `json:"IDAfijo"`
IDSexo     int    `json:"IDSexo"`
IDPerroMadre int   `json:"IDPerroMadre"`
IDPerroPadre int   `json:"IDPerroPadre"`
IDRaza      int    `json:"IDRaza"`
FechaNacimiento string `json:"FechaNacimiento"`
FechaDefuncion string `json:"FechaDefuncion"`
FechaAlta    string `json:"FechaAlta"`
FechaBaja    string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

cargarDatosIniciales_Propietarios

Registrar en el sistema los propietarios de los ejemplares definidos en un fichero de texto con el siguiente formato JSON:

```
type PerrosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int   `json:"IDPerroPropietario"`
    IDPerro          int    `json:"IDPerro"`
    IDPersona        int    `json:"IDPersona"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

consultarDatosEjemplar

Consultar los datos del ejemplar. La información presentada variará dependiendo de la persona que realiza la consulta.

obtenerCertificadoRegistro

Obtener un certificado con los datos del registro que certifica su inscripción en el libro genealógico.

obtenerPedigri

Obtener un fichero digital que contiene los datos del ejemplar y su línea genealógica que certifica su pureza de raza.

registrarCesionTemporal

Solicitar el registro de cesión temporal de la propiedad a uno o varios criadores por un tiempo limitado.

Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**





Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# PERROS
peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\": [{"\\Nombre\\":\\"PERRO_0001\\",\\"IDSexo\\":1},{\\\"Nombre\\":\\"PERRA_0001\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":181,\\"IDPerroPadre\\":193,\\"FechaNacimiento\\":\\"2020-08-23\\", {"\\IDPersona\\":6}]}]' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\": [{"\\Nombre\\":\\"PERRO_0002\\",\\"IDSexo\\":1},{\\\"Nombre\\":\\"PERRA_0002\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":0,\\"IDPerroPadre\\":0,\\"FechaNacimiento\\":\\"2020-08-23\\",\\"Propietarios\\": [{"\\IDPersona\\":600}, {"\\IDPersona\\":601}]], {"\\IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\": [{"\\Nombre\\":\\"PERRO_0001\\",\\"IDSexo\\":1},{\\\"Nombre\\":\\"PERRA_0001\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":0,\\"IDPerroPadre\\":193,\\"FechaNacimiento\\":\\"2020-08-23\\",\\"Propietarios\\": [{"\\IDPersona\\":600}, {"\\IDPersona\\":601}]], {"\\IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarCambioPropietario","Args": ["{\\"IDPerro\\":5760,\\"Propietarios\\": [{"\\IDPersona\\":401}, {"\\IDPersona\\":402}, {"\\IDPersona\\":403}, {"\\IDPersona\\":404}]}], {"\\IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarDefuncionPerro","Args": ["{\\"IDPerro\\":100,\\"FechaDefuncion\\":\\"2020-08-08\\"}, {"\\IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n perros -c
'{"function":"registrarReconocimientoRaza","Args": ["{\\"IDPerro\\":37,\\"IDRaza\\":126,\\"Justificacion\\":\\"Ha sido reconocida su raza por el Juez PEPE GUTIERREZ DOSSANTOS\\"}, {"\\IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

ERRORES PERROS

```
peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\":{\\"Nombre\\":\\"PERRO_0001\\",\\"IDSexo\\":1},{\\"Nombre\\":\\"PERRA_0001\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":181,\\"IDPerroPadre\\":3,\\"FechaNacimiento\\":\\"2020-08-23\\", {\\"IDPersona\\":6}]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\":{\\"Nombre\\":\\"PERRO_0001\\",\\"IDSexo\\":1},{\\"Nombre\\":\\"PERRA_0001\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":181,\\"IDPerroPadre\\":10,\\"FechaNacimiento\\":\\"2020-08-23\\", {\\"IDPersona\\":6}]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n perros -c
'{"function":"registrarPerro","Args":["{\\"Perros\\":{\\"Nombre\\":\\"PERRO_0001\\",\\"IDSexo\\":1},{\\"Nombre\\":\\"PERRA_0001\\",\\"IDSexo\\":0}],\\"IDPerroMadre\\":0,\\"IDPerroPadre\\":193,\\"FechaNacimiento\\":\\"2020-08-23\\", {\\"IDPersona\\":6}]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarCambioPropietario","Args":["{\\"IDPerro\\":577777777760,\\"Propietarios\\":{\\"IDPersona\\":401},{\"IDPersona\\":402},{\"IDPersona\\":403},{\"IDPersona\\":404}}", {\\"IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarCambioPropietario","Args":["{\\"IDPerro\\":5760,\\"Propietarios\\":{\\"IDPersona\\":401},{\"IDPersona\\":402},{\"IDPersona\\":44444444403},{\"IDPersona\\":404}}", {\\"IDPersona\\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```





```
Peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarDefuncionPerro","Args":[{"\"IDPerro\":100,\"FechaDefuncion\":
"\\"2020-08-08\""}, {"\"IDPersona\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE
-C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarDefuncionPerro","Args":[{"\"IDPerro\":1000000000,\"FechaDefuncion\":
"\\"2020-08-08\""}, {"\"IDPersona\":6}]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERROS -c
'{"function":"registrarDefuncionPerro","Args":[{"\"IDPerro\":100\"FechaDefuncion\":
"\\"2020-08-08\""}, {"\"IDPersona\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE
-C $CHANNEL_NAME

peer chaincode invoke -n perros -c
'{"function":"registrarReconocimientoRaza","Args":[{"\"IDPerro\":12,\"IDRaza\":126
,"\"Justificacion\":\"Ha sido reconocida su raza por el Juez PEPE GUTIERREZ
DOSSANTOS\"}, {"\"IDPersona\":6}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

CARGAS INICIALES

```
peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales","Args":["./json/perros_001.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales","Args":["./json/perros_002.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales","Args":["./json/perros_003.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales","Args":["./json/perros_004.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales","Args":["./json/perros_005.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n perros -c
'{"function":"asignarEstado","Args":["PERROS",
"\\"docType\":\"CONTADOR\",\"IDMaximo\":5759}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietari
os_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s
```



```
peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_002.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_003.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_004.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s

peer chaincode invoke -n perros -c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/perros_propietarios_005.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
sleep 7s

peer chaincode invoke -n perros -c
'{"function":"asignarEstado","Args":["PERROS_PROPIETARIOS",
"\\"docType\":"\"CONTADOR\",\"IDMaximo\":5759}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME
```



SOLICITUDES

Este contrato inteligente es el encargado de gestionar las solicitudes de los usuarios cuando es necesario realizar la validación o autorización de la acción por más de una persona.

Por ejemplo:

- Cuando se desea realizar un cambio de propietario de un ejemplar y este pertenece a varias personas, requiere la autorización de ambas personas.
- Cuando se desea registrar el nacimiento de una nueva camada y los progenitores pertenecen a dos criadores o propietarios distintos, no basta con que se registre la solicitud por parte de uno de ellos sino de ambos.



La solicitud es registrada en el sistema por cualquiera de los usuarios implicados, quedando pendiente su ejecución hasta que todos los implicados validan o rechazan el contenido de la solicitud o caduca por transcurrir un tiempo desde su registro.

Una vez que todos los implicados han validado la solicitud esta se ejecutara automáticamente.

En el caso en el que solo sea necesaria la intervención de un usuario y sea este mismo usuario el que registrara la solicitud, esta se ejecutar a también automáticamente.

Con estas certificaciones se evitan fraudes que son comunes, como por ejemplo vender un perro a un nuevo propietario falsificando su origen genealógico (por ejemplo, es hijo del campeón del España)



Las funciones que incorpora el contrato inteligente son similares a las vistas anteriormente en otros contratos:

- **Argumentos entrada:**

Serán los mismos parámetros de entrada que tendría la llamada a la función si realizara la acción directamente.

Por ejemplo:

solicitarRegistrarPerro tendrá los mismos parámetros de entrada la función **registrarPerro** del contrato **PERROS**:

```
type tipoRegistrarCamadaPerro struct {
    Nombre string `json:"Nombre"`
    IDSexo int `json:"IDSexo"`
}

type tipoRegistrarCamadaPropietario struct {
    IDPersona int `json:"IDPersona"`
}

type tipoRegistrarCamada struct {
    Perros []tipoRegistrarCamadaPerro `json:"Perros"`
    IDPerroMadre int `json:"IDPerroMadre"`
    IDPerroPadre int `json:"IDPerroPadre"`
    IDAfijo int `json:"IDAfijo"`
    IDRaza int `json:"IDRaza"`
    FechaNacimiento string `json:"FechaNacimiento"`
    Propietarios []tipoRegistrarCamadaPropietario `json:"Propietarios"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

- **Argumentos salida:**

Si la solicitud se puede ejecutar automáticamente devolverá los mismos parámetros de salida que la función **registrarPerro** del contrato **PERROS**:

```
type Perros struct {
    ObjectType string `json:"docType"`
    IDPerro int `json:"IDPerro"`
    Nombre string `json:"Nombre"`
    IDAfijo int `json:"IDAfijo"`
    IDSexo int `json:"IDSexo"`
}
```

```
IDPerroMadre    int      `json:"IDPerroMadre"`
IDPerroPadre    int      `json:"IDPerroPadre"`
IDRaza          int      `json:"IDRaza"`
FechaNacimiento string  `json:"FechaNacimiento"`
FechaDefuncion  string  `json:"FechaDefuncion"`
FechaAlta        string  `json:"FechaAlta"`
FechaBaja        string  `json:"FechaBaja"`
}

type PerrosPropietarios struct {
    ObjectType      string `json:"docType"`
    IDPerroPropietario int   `json:"IDPerroPropietario"`
    IDPerro          int   `json:"IDPerro"`
    IDPersona        int   `json:"IDPersona"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```



En caso contrario devolverá los registros de la solicitud insertados:

```
type Solicitudes struct {
    ObjectType      string `json:"docType"`
    IDSolicitud    int   `json:"IDSolicitud"`
    TipoSolicitud  string `json:"TipoSolicitud"`
    JSONSolicitud  string `json:"JSONSolicitud"`
    IDPersonaSolicitante int   `json:"IDPersonaSolicitante"`
    EstadoSolicitud string `json:"EstadoSolicitud"`
    FechaEjecucion  string `json:"FechaEjecucion"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}

type SolicitudesAutorizaciones struct {
    ObjectType      string `json:"docType"`
    IDSolicitud    int   `json:"IDSolicitud"`
    IDPersona        int   `json:"IDPersona"`
    EstadoSolicitud string `json:"EstadoSolicitud"
```



```
FechaEjecucion string `json:"FechaEjecucion"`
FechaAlta      string `json:"FechaAlta"`
FechaBaja      string `json:"FechaBaja"`
}
```

donde la autorización del usuario que realiza la solicitud estará registrada y aprobada automáticamente, por lo que no necesitará validar esta acción.

El contrato valida los mismos parámetros que si realizara la acción y no permitirá que se registren una solicitud si existen otra activa, referente a la misma acción.

De esta forma existe la siguiente correlación de funciones y parámetros entre contratos:

PERROS

solicitarRegistrarCamada

- Función **registrarPerro** del contrato **PERROS**

solicitarRegistrarPerro

- Función **registrarPerro** del contrato **PERROS**

solicitarRegistrarCambioPropietarioPerro

- Función **registrarCambioPropietario** del contrato **PERROS**

AFIJOS

solicitarRegistrarCambioPropietarioAfijo

- Función **registrarCambioPropietario** del contrato **AFIJOS**

solicitarRegistrarCancelacionAfijo

- Función **registrarCancelacionAfijo** del contrato **AFIJOS**



El contrato inteligente también incorpora las siguientes funciones:

validarSolicitud

Registrar las validaciones que los usuarios implicados en una solicitud realizan sobre una petición. En el sistema el número o código de microchip insertado subcutáneamente al ejemplar por un veterinario.

En el caso de que todos los usuarios implicados en la solicitud den el visto bueno a solicitud, esta se procederá a ejecutarse automáticamente.

- **Argumentos entrada:**

```
type tipoValidarSolicitud struct {
    IDSolicitud int `json:"IDSolicitud"`
    EstadoSolicitud string `json:"EstadoSolicitud"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido



- El IDSolicitud tenga un valor y corresponda a una solicitud que se encuentre en estado pendiente (no ejecutada y no caducada)
- El EstadoSolicitud tenga un valor valido (APROBADO / RECHAZADO).
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

- **Argumentos salida:**

En el caso de que el usuario valide la solicitud y no quede ningún otro usuario por validar el contrato ejecutara automáticamente la solicitud devolviendo los parámetros de la ejecución de la acción solicitada.

En caso contrario devolverá el registros actualizado:

```
type SolicitudesAutorizaciones struct {
    ObjectType      string `json:"docType"`
    IDSolicitud     int    `json:"IDSolicitud"`
    IDPersona        int    `json:"IDPersona"`
    EstadoSolicitud string `json:"EstadoSolicitud"`
    FechaEjecucion  string `json:"FechaEjecucion"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```

cargarDatosIniciales

Introduce en el sistema el historial de las solicitudes que han realizado los diferentes usuarios, definidos en un fichero de texto con el siguiente formato JSON:

```
type Solicitudes struct {
    ObjectType      string `json:"docType"`
    IDSolicitud     int    `json:"IDSolicitud"`
    TipoSolicitud   string `json:"TipoSolicitud"`
    JSONSolicitud   string `json:"JSONSolicitud"`
    IDPersonaSolicitante int `json:"IDPersonaSolicitante"`
    EstadoSolicitud string `json:"EstadoSolicitud"`
    FechaEjecucion  string `json:"FechaEjecucion"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```



- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```



cargarDatosIniciales_Autorizaciones

Introduce en el sistema el historial de las autorizaciones de las solicitudes que han realizado los diferentes usuarios, definidos en un fichero de texto con el siguiente formato JSON:

```
type SolicitudesAutorizaciones struct {
    ObjectType      string `json:"docType"`
    IDSolicitud    int    `json:"IDSolicitud"`
    IDPersona       int    `json:"IDPersona"`
    EstadoSolicitud string `json:"EstadoSolicitud"`
    FechaEjecucion string `json:"FechaEjecucion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```



- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

Funciones complementarias

- **querySolicitudes**

Permite a un usuario consultar las solicitudes de un IDPersona y sus estados

Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**

Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# SOLICITUDES

# solicitarRegistrarCamada / Perro
peer chaincode invoke -n solicitudes -c
'{"function":"solicitarRegistrarCamada","Args":[{"\"Perros\":[{\"nombre\":\"PERRO_"
"uno\", \"IDSexo\":1}, {"\"nombre\":\"PERRA_"
"uno\", \"IDSexo\":0}], \"IDPerroMadre\":0, \""
"
```

```
IDPerroPadre\":0,\\"FechaNacimiento\":\\"2020-08-  
23\",\\\"Propietarios\":[{\\"IDPersona\":666},{\\\"IDPersona\":999}]}},  
\"{\\\"IDPersona\":6}"]}]' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n solicitudes -c  
'{"function":"solicitarRegistrarCamada","Args":["{\\"Perros\":[{\\"nombre\":\"\\PERRO_  
dos\",\\\"IDSexo\":1},{\"nombre\":\"\\PERRA_dos\",\\\"IDSexo\":1}],\\\"IDPerroMadre\":1,\\\"  
IDPerroPadre\":50,\\\"FechaNacimiento\":\\"2020-08-23\\\"}},{\\\"IDPersona\":6}"]}' -o  
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n solicitudes -c  
'{"function":"ejecutarConsulta","Args":["{\\"selector\": {\\"docType\\":  
\\\"SOLICITUDES_AUTORIZACIONES\\\", \\"EstadoSolicitud\\\": \\"PENDIENTE\\\", \\"FechaBaja\\\":  
{ \\"$lt\\\": \\"2020-09-15\\\"}} }, {\\\"IDPersona\":5}"]}' -o $ORDENER_URL --tls --  
cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n solicitudes -c  
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\":2,\\\"EstadoSolicitud\\\":\\\"  
APROBADO\\\"},{\\\"IDPersona\":5}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C  
$CHANNEL_NAME  
peer chaincode invoke -n solicitudes -c  
'{"function":"solicitarRegistrarCamada","Args":["{\\"Perros\":[{\\"nombre\":\"\\PADRE_  
tres\",\\\"IDSexo\":1},{\"nombre\":\"\\MADRE_tres\",\\\"IDSexo\":1}],\\\"IDPerroMadre\":1,  
\\\"IDPerroPadre\":50,\\\"FechaNacimiento\":\\"2021-05-22\\\"}},{\\\"IDPersona\":6}"]}' -  
o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME  
peer chaincode invoke -n solicitudes -c  
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\":3,\\\"EstadoSolicitud\\\":\\\"  
RECHAZADO\\\"},{\\\"IDPersona\":5}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C  
$CHANNEL_NAME
```





solicitarRegistrarCambioPropietarioPerro

```
peer chaincode invoke -n $CC_NOMBRE_SOLICITUDES -c
'{"function":"solicitarRegistrarCambioPropietarioPerro","Args":["{\\"IDPerro\\":1,\\"Propietarios\\":[\{\\"IDPersona\\":1\}, {\\"IDPersona\\":2}\}], {\\"IDPersona\\":1}"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\\":4,\\"EstadoSolicitud\\":\\"APROBADO\\"}, {\\"IDPersona\\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

solicitarRegistrarCambioPropietarioAfijo

```
peer chaincode invoke -n solicitudes -c
'{"function":"solicitarRegistrarCambioPropietarioAfijo","Args":["{\\"IDAfijo\\":2,\\"Propietarios\\":[\{\\"IDPersona\\":201\}, {\\"IDPersona\\":202\}, {\\"IDPersona\\":203\}, {\\"IDPersona\\":204}\}], {\\"IDPersona\\":5}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

```
peer chaincode invoke -n solicitudes -c
'{"function":"solicitarRegistrarCambioPropietarioAfijo","Args":["{\\"IDAfijo\\":2,\\"Propietarios\\":[\{\\"IDPersona\\":5\}]\}, {\\"IDPersona\\":201}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

```
peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\\":6,\\"EstadoSolicitud\\":\\"APROBADO\\"}, {\\"IDPersona\\":202}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

```
peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\\":6,\\"EstadoSolicitud\\":\\"APROBADO\\"}, {\\"IDPersona\\":203}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

```
peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\\":6,\\"EstadoSolicitud\\":\\"APROBADO\\"}, {\\"IDPersona\\":204}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

solicitarRegistrarCancelacionAfijo"

```
peer chaincode invoke -n solicitudes -c
'{"function":"solicitarRegistrarCancelacionAfijo","Args":["{\\"IDAfijo\\":50\}, {\\"IDPersona\\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n solicitudes -c
'{"function":"solicitarRegistrarCambioPropietarioAfijo","Args":["{\\"IDAfijo\\":3,\\"Propietarios\\":[\{\\"IDPersona\\":301\}, {\\"IDPersona\\":302}\}], {\\"IDPersona\\":123}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

ERRORES SOLICITUDES

```
peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":["{\\"IDSolicitud\\":2,\\"EstadoSolicitud\\":\\"APROBADO\\"}, {\\"IDPersona\\":555}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```



```
peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":[{"\"IDSolicitud\":2,\"EstadoSolicitud\":\"APROBADO\""}, {"\"IDPersona\":5}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_SOLICITUDES -c
'{"function":"solicitarRegistrarCambioPropietarioPerro","Args": [{"\"IDPerro\":1234
56,\"Propietarios\":[{\\"IDPersona\":1},{\"IDPersona\":2}]}",
"\"IDPersona\":1}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

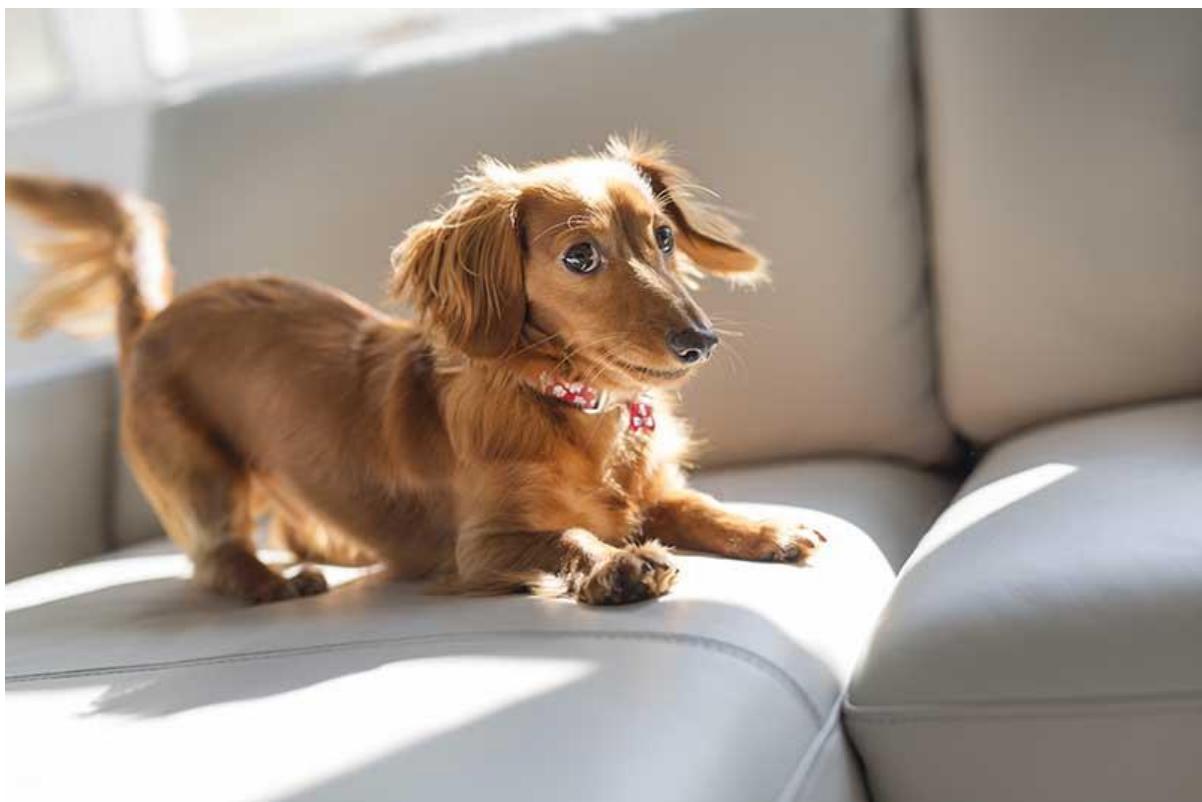
peer chaincode invoke -n $CC_NOMBRE_SOLICITUDES -c
'{"function":"solicitarRegistrarCambioPropietarioPerro","Args": [{"\"IDPerro\":1234
56,\"Propietarios\":[{\\"IDPersona\":1765},{\"IDPersona\":23456}]}",
"\"IDPersona\":1234}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n solicitudes -c
'{"function":"validarSolicitud","Args":[{"\"IDSolicitud\":7,\"EstadoSolicitud\":\"APROBADO\""}, {"\"IDPersona\":259}]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME
```

CARGAS INICIALES

```
peer chaincode invoke -n solicitudes-c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/solicitudes.json"
]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n solicitudes-c
'{"function":"cargarDatosIniciales_Propietarios","Args":["./json/autorizaciones.js
on"]}'' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```



MICROCHIP

Registro de identificación de microchip subcutáneo insertado por los veterinarios que contiene un transpondedor con un código único que permite identificar de manera unívoca al ejemplar.



Funciones que incorpora el contrato inteligente:

registrarMicrochips

Introduce en el sistema el número o código de microchip insertado subcutáneamente al ejemplar por un veterinario

- **Argumentos entrada:**

```
type MicrochipsPerros struct {
    IDPerro          int    `json:"IDPerro"`
    IDPersonaVeterinario int   `json:"IDPersonaVeterinario"`
    CODMicrochip     string `json:"CODMicrochip"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```



El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- Los argumentos Nombre y Propietarios tenga un valor.
- El IDPerro tenga un valor y corresponda a un ejemplar vivo.
- El IDPersonaVeterinario tenga un valor y corresponda a una persona con un perfil de Veterinario activo.
- El CODMicrochip tenga un valor
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

• **Argumentos salida:**

Devuelve los registros insertados:

```
type MicrochipsPerros struct {  
    ObjectType      string `json:"docType"  
    IDMicrochipPerro int   `json:"IDMicrochipPerro"  
    IDPerro         int   `json:"IDPerro"  
    IDPersonaVeterinario int  `json:"IDPersonaVeterinario"  
    CODMicrochip    string `json:"CODMicrochip"  
    FechaAlta       string `json:"FechaAlta"  
    FechaBaja       string `json:"FechaBaja"  
}
```

consultarMicrochip

Consultar los datos del microchip y del ejemplar. La información presentada variara dependiendo de la persona que realiza la consulta.

cargarDatosIniciales

Introduce en el sistema el historial de inserciones de microchips que han tenido los diferentes ejemplares, definidos en un fichero de texto con el siguiente formato JSON:

```
type MicrochipsPerros struct {  
    ObjectType      string `json:"docType"  
    IDMicrochipPerro int   `json:"IDMicrochipPerro"  
    IDPerro         int   `json:"IDPerro"  
    IDPersonaVeterinario int  `json:"IDPersonaVeterinario"  
}
```



```
CODMicrochip      string `json:"CODMicrochip"`
FechaAlta        string `json:"FechaAlta"`
FechaBaja        string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```



Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**

Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente



```
export CHANNEL_NAME=netcanchannel

export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem

export ORDENER_URL=orderer.netcan.com:7050

# MICROCHIPS

peer chaincode invoke -n $CC_NOMBRE_MICROCHIPS -c
'{"function":"registrarMicrochipPerro","Args":[{"\"IDPerro\":200,\"IDPersonaVeterinario\":100,\"CODMicrochip\":\"123456789012345\"},{\"\"IDPersona\":6}"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# ERRORES MICROCHIPS

peer chaincode invoke -n $CC_NOMBRE_MICROCHIPS -c
'{"function":"registrarMicrochipPerro","Args":[{"\"IDPerro\":200,\"IDPersonaVeterinario\":1000000000000,\"CODMicrochip\":\"123456789012345\"},{\"\"IDPersona\":6}"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_MICROCHIPS -c
'{"function":"registrarMicrochipPerro","Args":[{"\"IDPerro\":200,\"IDPersonaVeterinario\":100,\"CODMicrochip\":\"123456789012345\"},{\"\"IDPersona\":6}"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# CARGAS INICIALES

peer chaincode invoke -n microchips -c
'{"function":"cargarDatosIniciales","Args":[("./json/microchips_perros_001.json")]}'
-o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n microchips -c
'{"function":"cargarDatosIniciales","Args":[("./json/microchips_perros_002.json")]}'
-o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n microchips -c
'{"function":"cargarDatosIniciales","Args":[("./json/microchips_perros_003.json")]}'
-o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n microchips -c
'{"function":"cargarDatosIniciales","Args":[("./json/microchips_perros_004.json")]}'
-o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n microchips -c
'{"function":"cargarDatosIniciales","Args":[("./json/microchips_perros_005.json")]}'
-o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n microchips -c
'{"function":"asignarEstado","Args":["MICROCHIPS_PERROS",
"\"docType\":\"CONTADOR\", \"IDMaximo\":5759}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME
```

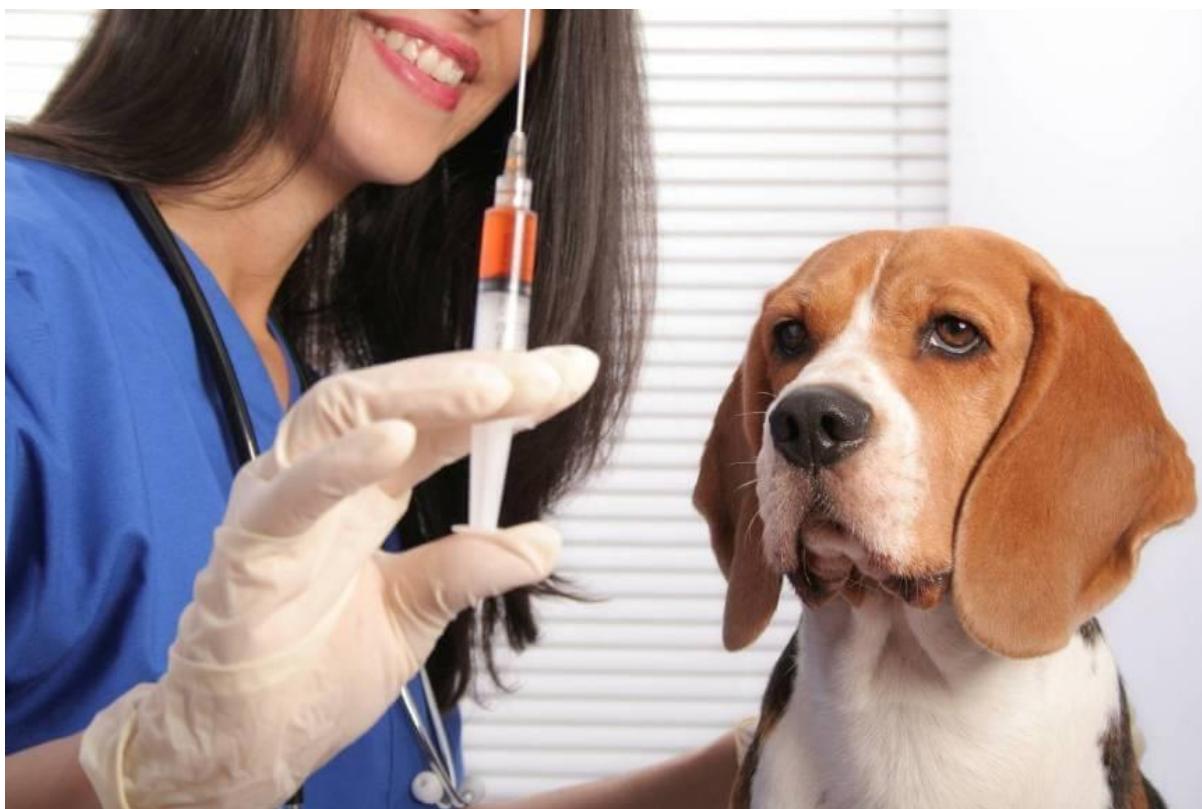


VACUNAS

Registro de vacunas administradas por veterinarios.

Los nombres de las vacunas suelen variar según el fabricante y aunque los nombres puedan ser similares (Pentavalente, Hexavalente u Octovalente) su contenido puede variar su contenido con vacunas contra diversas enfermedades tales como, por ejemplo:

- el moquillo canino,
- hepatitis infecciosa
- la leptospirosis
- el parvovirus
- el coronaviru
- la rabia
- la parainfluenza
- la Bordetella bronchisepticala
- la borreliosis o enfermedad de Lyme
- la herpesvirus canino
- la leishmaniasis





Funciones que incorpora el contrato inteligente:

registrarVacuna

Registrar en el sistema el tipo de vacuna administrada al ejemplar, su identificador medicinal, el identificador del colegiado del veterinario, la fecha de administración y la duración de la dosis.

Introduce en el sistema el número o código de microchip insertado subcutáneamente al ejemplar por un veterinario

- **Argumentos entrada:**

```
type tipoRegistrarVacunaPerroPropietario struct {  
    IDVacunaProteccion int `json:"IDProteccion"  
    FechaBaja string `json:"FechaBaja"  
}  
  
type tipoRegistrarVacunaPerro struct {  
    IDPerro int `json:"IDPerro"  
    IDPersonaVeterinario int `json:"IDPersonaVeterinario"  
    CODVacuna string `json:"CODVacuna"  
    FechaAlta string `json:"FechaAlta"  
    FechaBaja string `json:"FechaBaja"  
    Protecciones []tipoRegistrarVacunaPerroPropietario `json:"Protecciones"  
}
```

```
type TipoSeguridad struct {  
    IDPersona int `json:"IDPersona"  
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- Los argumentos CODVacuna, FechaAlta y Protecciones tenga un valor.
- El IDPerro tenga un valor y corresponda a un ejemplar vivo.
- El IDPersonaVeterinario tenga un valor y corresponda a una persona con un perfil de Veterinario activo.
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

- **Argumentos salida:**

Devuelve los registros insertados:

```
type MicrochipsPerros struct {  
    ObjectType      string `json:"docType"``  
    IDMicrochipPerro int   `json:"IDMicrochipPerro"``  
    IDPerro          int   `json:"IDPerro"``  
    IDPersonaVeterinario int  `json:"IDPersonaVeterinario"``  
    CODMicrochip    string `json:"CODMicrochip"``  
    FechaAlta       string `json:"FechaAlta"``  
    FechaBaja       string `json:"FechaBaja"``  
}
```

obtenerCertificadoVacunaciones (pasaporte)

Obtener un fichero digital que contiene los datos del ejemplar y su cartilla de vacunación.

consultarVacunaciones

Consultar la cartilla de vacunación del ejemplar. La información presentada variara dependiendo de la persona que realiza la consulta.





cargarDatosIniciales

Introduce en el sistema el historial de vacunaciones que han tenido los diferentes ejemplares, definidos en un fichero de texto con el siguiente formato JSON:

```
type VacunasPerros struct {
    ObjectType      string `json:"docType"`
    IDVacunaPerro  int   `json:"IDVacunaPerro"`
    IDPerro         int   `json:"IDPerro"`
    IDPersonaVeterinario int `json:"IDPersonaVeterinario"`
    CODVacuna       string `json:"CODVacuna"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

cargarDatosIniciales_VacunasPerrosProteccion

Introduce en el sistema el historial de vacunaciones que han tenido los diferentes ejemplares, definidos en un fichero de texto con el siguiente formato JSON:

```
type VacunasPerrosProteccion struct {
    ObjectType      string `json:"docType"`
    IDVacunaPerroProteccion int `json:"IDVacunaPerroProteccion"`
    IDVacunaPerro  int   `json:"IDVacunaPerro"`
    IDVacunaProteccion int `json:"IDVacunaProteccion"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```



- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

cargarDatosIniciales_VacunasProteccion

Introduce en el sistema el repositorio de tipos de vacunas más comunes, que determinan el tipo de protección que contiene cada vacuna, que han tenido los diferentes ejemplares, definidos en un fichero de texto con el siguiente formato JSON:

```
type VacunasProteccion struct {
    ObjectType      string `json:"docType"`
    IDVacunaProteccion int `json:"IDVacunaProteccion"`
    Nombre          string `json:"Nombre"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```





Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**

Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# VACUNAS
peer chaincode invoke -n vacunas -c '{"function":"registrarVacunaPerro", "Args":["\\"IDPerro\":6, \\"IDPersonaVeterinario\":106, \\"CODVacuna\":\\"123456789012345\\", \\"FechaBaja\":\\"2022-02-02\\", \\"Protecciones\":[\\"IDVacunaProteccion\":1,\\"FechaBaja\":\\"2022-02-02\\", {\\"IDVacunaProteccion\":2,\\"FechaBaja\":\\"2022-02-02\\"} ]}, {\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# ERRORES VACUNAS
peer chaincode invoke -n vacunas -c '{"function":"registrarVacunaPerro", "Args":["\\"IDPerro\":6, \\"IDPersonaVeterinario\":106, \\"CODVacuna\":\\"123456789012345\\", \\"FechaBaja\":\\"2022-02-02\\", \\"Protecciones\":[\\"IDVacunaProteccion\":1,\\"FechaBaja\":\\"2022-02-02\\", {\\"IDVacunaProteccion\":2,\\"FechaBaja\":\\"2022-02-02\\"} ]}, {\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
peer chaincode invoke -n vacunas -c '{"function":"registrarVacunaPerro", "Args":["\\"IDPerro\":6, \\"IDPersonaVeterinario\":106, \\"CODVacuna\":\\"987654321012345\\", \\"FechaBaja\":\\"2022-02-02\\", \\"Protecciones\":[\\"IDVacunaProteccion\":1,\\"FechaBaja\":\\"2022-02-02\\", {\\"IDVacunaProteccion\":230,\\"FechaBaja\":\\"2022-02-02\\"} ]}, {\\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```



CARGAS INICIALES

```
peer chaincode invoke -n vacunas -c
'{"function":"cargarDatosIniciales_VacunasProteccion","Args":["./json/vacunas_prot
ecction_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n vacunas -c
'{"function":"asignarEstado","Args":["VACUNAS_PERROS_PROTECCION",
"\\"docType\":"CONTADOR\","IDMaximo":512}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n vacunas -c
'{"function":"cargarDatosIniciales","Args":["./json/vacunas_perros_001.json"]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

sleep 7s

peer chaincode invoke -n vacunas -c
'{"function":"asignarEstado","Args":["VACUNAS_PROTECCION",
"\\"docType\":"CONTADOR\","IDMaximo":11}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n vacunas -c
'{"function":"cargarDatosIniciales_VacunasPerrosProteccion","Args":["./json/vacuna
s_perros_proteccion_001.json"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C
$CHANNEL_NAME

sleep 7s

peer chaincode invoke -n vacunas -c
'{"function":"asignarEstado","Args":["VACUNAS_PERROS",
"\\"docType\":"CONTADOR\","IDMaximo":89}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME
```



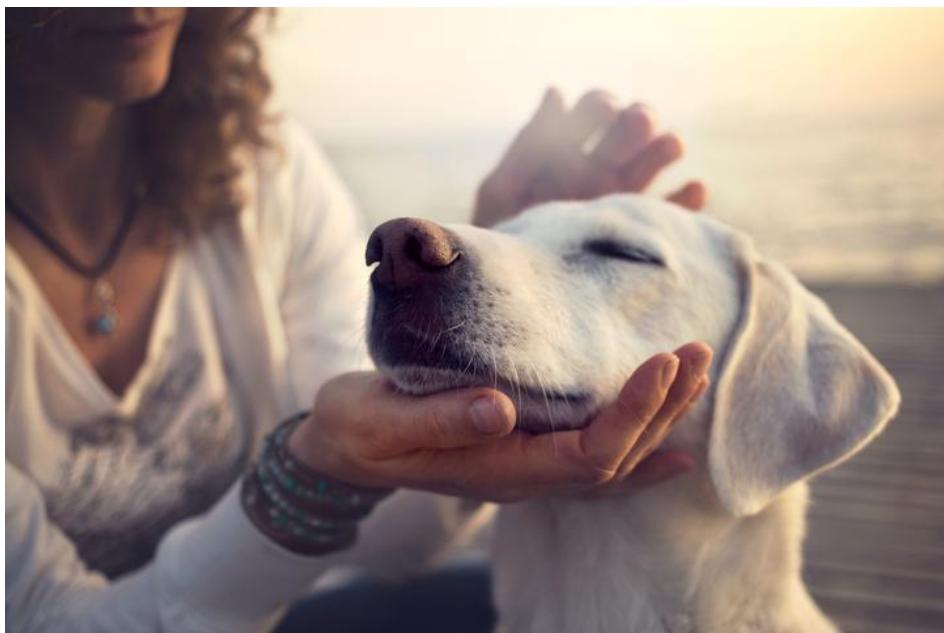


PERFILES

Contrato que gestiona el mantenimiento de los diferentes roles especiales de usuario:

- Administradores
- Veterinarios
- Trabajadores de las federaciones caninas

La asignación de los perfiles puede ser consultado desde cualquier contrato, pero solo podrán ser modificados por administradores.



Funciones que incorpora el contrato inteligente:

registrarPerfilPersona

Inserta en el sistema la definición de un nuevo perfil para un usuario.

- **Argumentos entrada:**

```
type PerfilesPersonasInsertar struct {
    IDPersona      int     `json:"IDPersona"`
    CODPerfil     string  `json:"CODPerfil"`
}
```



```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPersona este registrado.
- El CODPerfil tenga un valor
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción (Perfil administrador)

- **Argumentos salida:**

Devuelve los registros insertados:

```
type PerfilesPersonas struct {
    ObjectType      string `json:"docType"`
    IDPerfilPersona int   `json:"IDPerfilPersona"`
    IDPersona       int   `json:"IDPersona"`
    CODPerfil       string `json:"CODPerfil"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

cancelarPerfilPersona

Cancela en el sistema la definición de un perfil existente para un usuario.

- **Argumentos entrada:**

```
type PerfilesPersonasCancelar struct {
    IDPersona      int   `json:"IDPersona"`
    CODPerfil     string `json:"CODPerfil"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```



El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPersona este registrado.
- El CODPerfil tenga un valor
- El IDPersona y CODPerfil tenga definido un perfil activo
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción (Perfil administrador)

• **Argumentos salida:**

Devuelve los registros insertados:

```
type PerfilesPersonas struct {  
    ObjectType      string `json:"docType"  
    IDPerfilPersona int   `json:"IDPerfilPersona"  
    IDPersona        int   `json:"IDPersona"  
    CODPerfil       string `json:"CODPerfil"  
    FechaAlta       string `json:"FechaAlta"  
    FechaBaja       string `json:"FechaBaja"  
}
```





cargarDatosIniciales

Introduce en el sistema el historial de perfiles que han tenido los diferentes usuarios, definidos en un fichero de texto con el siguiente formato JSON:

```
type PerfilesPersonas struct {
    ObjectType      string `json:"docType"`
    IDPerfilPersona int   `json:"IDPerfilPersona"`
    IDPersona       int   `json:"IDPersona"`
    CODPerfil      string `json:"CODPerfil"`
    FechaAlta      string `json:"FechaAlta"`
    FechaBaja      string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON args[1] sea valido
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción (Perfil administrador)

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

Funciones complementarias

- **asignarEstado_OnlyAdmin**

Permite a un usuario administrador modificar un estado de la cadena de bloques.

- **borrarEstado_OnlyAdmin**

Permite a un usuario administrador modificar un estado de la cadena de bloques

- **esAdministrador**

Indica si la persona consultada dispone de un perfil de Administrador activo.

- **esMiembroFederacion**

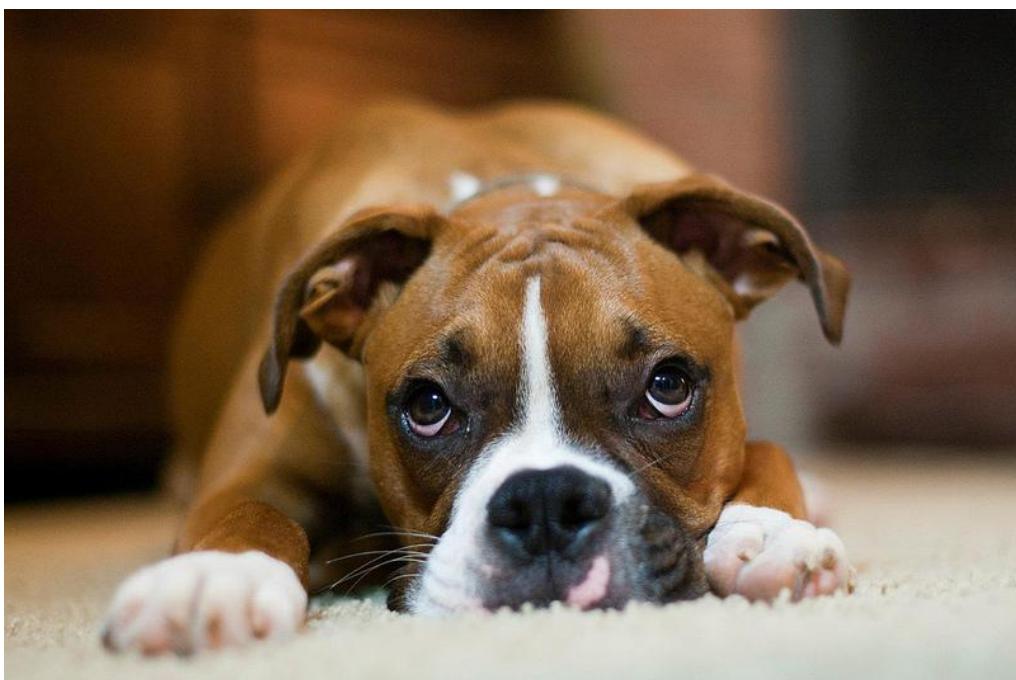
Indica si la persona consultada dispone de un perfil de miembro en una Federación Canina activo.

- **esVeterinario**

Indica si la persona consultada dispone de un perfil de Veterinario activo.

- **esPerfilPersona**

Indica si la persona consultada dispone de un perfil específico activo.



Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**



Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem
export ORDENER_URL=orderer.netcan.com:7050

# PERFILES
peer chaincode invoke -n $CC_NOMBRE_PERFILES -c
'{"function":"registrarPerfilPersona","Args":[{"\\"IDPersona\\":100,\\"CODPerfil\\":\\"ADMINISTRADOR\\"}], {"\\"IDPersona\\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERFILES -c
'{"function":"cancelarPerfilPersona","Args":[{"\\"IDPersona\\":100,\\"CODPerfil\\":\\"ADMINISTRADOR\\"}, {"\\"IDPersona\\":100}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# ERRORES PERFILES
peer chaincode invoke -n perfiles -c
'{"function":"cargarDatosIniciales","Args":[("./json/perfiles.json",
"\\"IDPersona\\":123456}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERFILES -c
'{"function":"cancelarPerfilPersona","Args":[{"\\"IDPersona\\":100,\\"CODPerfil\\":\\"ADMINISTRADOR\\"}, {"\\"IDPersona\\":6}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n $CC_NOMBRE_PERFILES -c
'{"function":"cancelarPerfilPersona","Args":[{"\\"IDPersona\\":666,\\"CODPerfil\\":\\"ADMINISTRADOR\\"}, {"\\"IDPersona\\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

# CARGAS INICIALES
peer chaincode invoke -n perfiles -c
'{"function":"cargarDatosIniciales","Args":[("./json/perfiles.json",
"\\"IDPersona\\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n perfiles -c
'{"function":"asignarEstado_OnlyAdmin","Args":["PERFILES_PERSONAS",
"\\"docType\\":\\"CONTADOR\\",\\"IDMaximo\\":48}, {"\\"IDPersona\\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n perfiles -c
'{"function":"cancelarPerfilPersona","Args":[{"\\"IDPersona\\":0,\\"CODPerfil\\":\\"ADMINISTRADOR\\"}, {"\\"IDPersona\\":0}"]}' -o $ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

RAZAS

Contrato que gestiona el mantenimiento de los diferentes estándares de pureza de raza reconocidos, actualmente cerca de 346 razas.

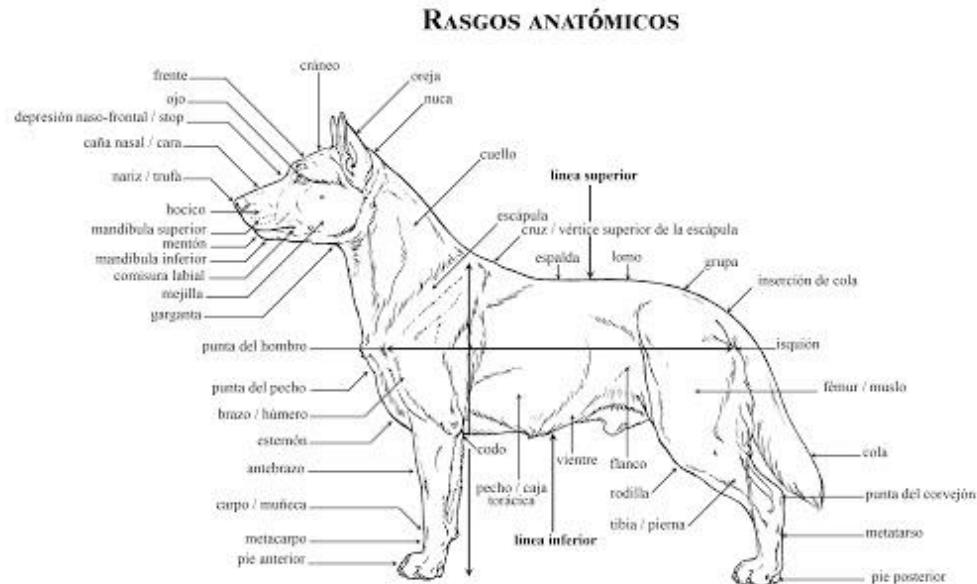
Se clasifican en los siguientes grupos:

- **Grupo 1:** Perros de pastor y perros boyeros (excepto perros boyeros suizos)
 - **Grupo 2:** Perros tipo pinscher y schnauzer, molosoides y perros tipo montaña y boyeros suizos
 - **Grupo 3:** Terriers
 - **Grupo 4:** Teckels
 - **Grupo 5:** Tipo spitz y tipo primitivo
 - **Grupo 6:** Perros tipo sabueso, perros de rastro (exceptuando lebreles) y razas semejantes.
 - **Grupo 7:** Perros de muestra
 - **Grupo 8:** Perros cobradores de caza - perros levantadores de caza - perros de agua
 - **Grupo 9:** Perros de compañía
 - **Grupo 10:** Lebreles



- **Razas Provisionales:** Razas provisionalmente aceptadas
- **Mestizos:** de padres desconocidos, diferente raza o cuyo origen de especie no esté validado.

La definición de las razas y grupos puede ser consultado desde cualquier contrato, pero solo podrán ser modificados por miembros de federaciones caninas.



Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**

Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
```



```
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizations/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-cert.pem

export ORDENER_URL=orderer.netcan.com:7050

# CARGAS INICIALES

peer chaincode invoke -n razas -c
'{"function":"cargarDatosIniciales_Grupos","Args":[("./json/grupos.json")]}' -o
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n razas -c '{"function":"asignarEstado","Args":["GRUPOS",
"\\"docType\":"\\CONTADOR\\",\\"IDMaximo\":12}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n razas -c
'{"function":"cargarDatosIniciales","Args":[("./json/razas.json")]}' -o $ORDENER_URL
--tls --cafile $CA_FILE -C $CHANNEL_NAME

peer chaincode invoke -n razas -c '{"function":"asignarEstado","Args":["RAZAS",
"\\"docType\":"\\CONTADOR\\",\\"IDMaximo\":346}"]}' -o $ORDENER_URL --tls --cafile
$CA_FILE -C $CHANNEL_NAME
```



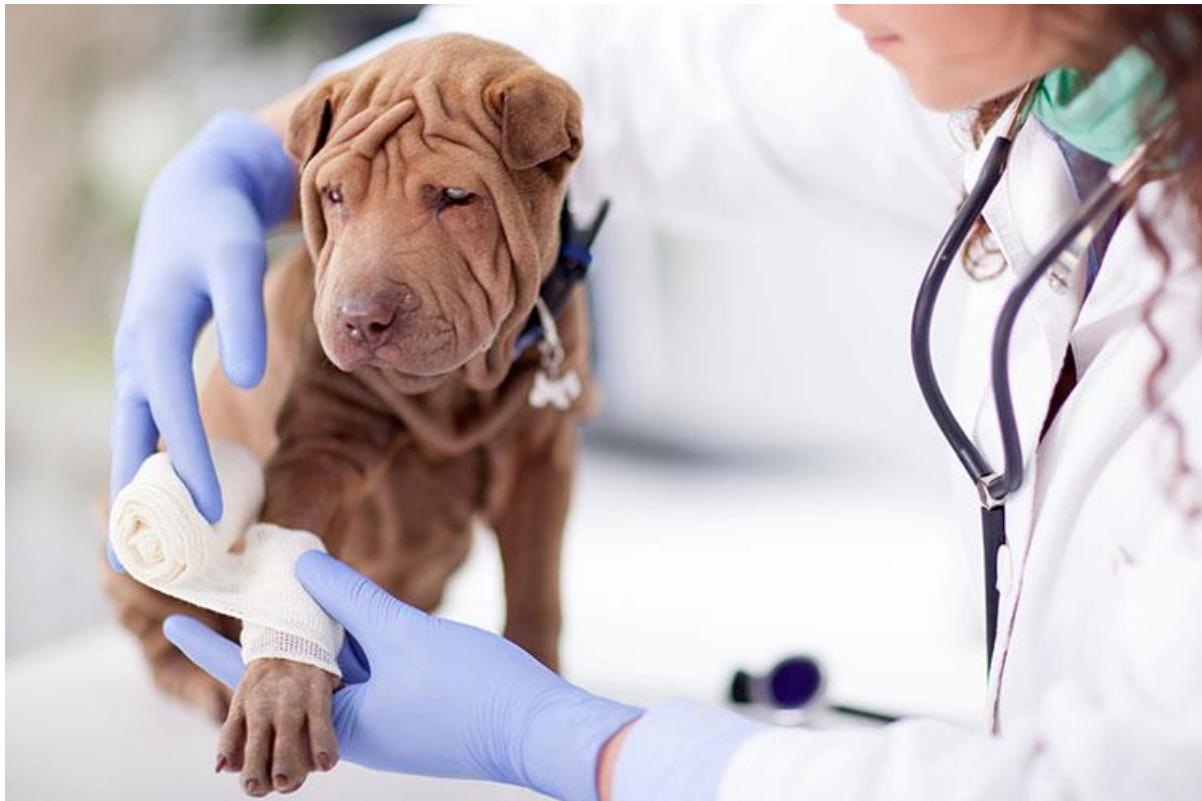


VETERINARIOS

Contrato que gestiona los datos específicos de un veterinario.

Para la prueba de concepto se ha simplificado los datos a gestionar reduciéndolos al número de colegiado.

En un caso real este contrato podrá contener toda la información que sea necesaria en relación con los veterinarios.



Funciones que incorpora el contrato inteligente:

registrarColegiaturaPersona

Inserta en el sistema el registro de una colegiatura de un usuario.

- **Argumentos entrada:**

```
type ColegiaturasPersonasRegistro struct {
    IDPersona          int    `json:"IDPersona"`
    CODColegiatura    string `json:"CODColegiatura"`
}
```



```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPersona este registrado.
- El CODColegiatura tenga un valor y no este registrado ya en el sistema
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

- **Argumentos salida:**

Devuelve los registros insertados:

```
type ColegiaturasPersonas struct {
    ObjectType      string `json:"docType"`
    IDColegiaturaPersona int `json:"IDColegiaturaPersona"`
    IDPersona       int `json:"IDPersona"`
    CODColegiatura  string `json:"CODColegiatura"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

cancelarPerfilPersona

Cancela en el sistema la definición de una colegiatura existente para un usuario.

- **Argumentos entrada:**

```
type PerfilesPersonasCancelar struct {
    IDPersona      int `json:"IDPersona"`
    CODPerfil     string `json:"CODPerfil"`
}
```

```
type TipoSeguridad struct {
    IDPersona int `json:"IDPersona"`
}
```



}

El contrato realiza las siguientes validaciones:

- El número de argumentos de entrada sea 2
- El formato JSON de args[0] y args[1] sea valido
- El IDPersona este registrado.
- El CODColegiatura tenga un valor
- El IDPersona y CODColegiatura tenga definido un perfil activo
- La persona que invoca la acción está registrada y dispone de los permisos para realizar dicha acción

• **Argumentos salida:**

Devuelve los registros insertados:

```
type ColegiaturasPersonas struct {
    ObjectType      string `json:"docType"`
    IDColegiaturaPersona int   `json:"IDColegiaturaPersona"`
    IDPersona        int   `json:"IDPersona"`
    CODColegiatura   string `json:"CODColegiatura"`
    FechaAlta        string `json:"FechaAlta"`
    FechaBaja        string `json:"FechaBaja"`
}
```





cargarDatosIniciales

Introduce en el sistema el historial de colegiaturas que han tenido los diferentes usuarios, definidos en un fichero de texto con el siguiente formato JSON:

```
type ColegiaturasPersonas struct {
    ObjectType      string `json:"docType"`
    IDColegiaturaPersona int `json:"IDColegiaturaPersona"`
    IDPersona       int `json:"IDPersona"`
    CODColegiatura  string `json:"CODColegiatura"`
    FechaAlta       string `json:"FechaAlta"`
    FechaBaja       string `json:"FechaBaja"`
}
```

- **Argumentos entrada:**

```
NombreFichero string
```

- **Argumentos salida:**

Devuelve el número de registros insertados:

```
NumeroRegistros string
```

Funciones comunes

Funciones de los contratos del sistema incorporadas a este contrato:

- **getQueryResultForQueryString**
- **asignarEstado**
- **borrarEstado**
- **consultarEstado**
- **consultarRangoEstados**
- **ejecutarConsulta**

Plan de pruebas

A continuación, se especifican una serie de comando que permite al lector realizar una batería de pruebas sobre el contrato inteligente

```
export CHANNEL_NAME=netcanchannel
export
CA_FILE=/opt/gopath/src/github.com/hyperledger/fabric/peer/crypto/ordererOrganizat
```



```
ions/netcan.com/orderers/orderer.netcan.com/msp/tlscacerts/tlsca.netcan.com-  
cert.pem
```

```
export ORDENER_URL=orderer.netcan.com:7050
```

VETERINARIOS

```
peer chaincode invoke -n veterinarios -c  
'{"function":"registrarColegiaturaPersona","Args":[{"\"IDPersona\":1,\"CODColegiatura\":\"COVM-1234567890\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"cancelarColegiaturaPersona","Args":[{"\"IDPersona\":1,\"CODColegiatura\":\"COVM-1234567890\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

ERRORES VETERINARIOS

```
peer chaincode invoke -n veterinarios -c  
'{"function":"registrarColegiaturaPersona","Args":[{"\"IDPersona\":1,\"CODColegiatura\":\"COVM-1234567890\"}, {"\"IDPersona\":123}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"registrarColegiaturaPersona","Args":[{"\"IDPersona\":126,\"CODColegiatura\":\"COVM-1234567890\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"registrarColegiaturaPersona","Args":[{"\"IDPersona\":10000000,\"CODColegiatura\":\"COVM-1234567890\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"cancelarColegiaturaPersona","Args":[{"\"IDPersona\":1,\"CODColegiatura\":\"COVM-NO-EXISTE\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"registrarColegiaturaPersona","Args":[{"\"IDPersona\":126,\"CODColegiatura\":\"1234567890\"}, {"\"IDPersona\":6}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```

CARGAS INICIALES

```
peer chaincode invoke -n veterinarios -c  
'{"function":"cargarDatosIniciales","Args":["./json/veterinarios.json"]}' -o  
$ORDENER_URL --tls --cafile $CA_FILE -C $CHANNEL_NAME
```

```
leep 7s
```

```
peer chaincode invoke -n veterinarios -c  
'{"function":"asignarEstado","Args":["VETERINARIOS_PERSONAS",  
"\\"docType\\":\\"CONTADOR\\",\\"IDMaximo\\":21}"]}' -o $ORDENER_URL --tls --cafile  
$CA_FILE -C $CHANNEL_NAME
```



Otros posibles contratos inteligentes

A continuación, se especifican algunos contratos más que podrían incorporarse en el futuro a la Blockchain, a modo de ejemplo:

TITULOS

Registra los títulos concedidos por federaciones, asociaciones y clubes por los méritos obtenidos por los ejemplares en explosiones y/o concursos

solicitarTitulo

Registra la solicitud de un título a una organización en base a los resultados obtenidos en exposiciones y concursos. La validación se realiza de manera automática por el sistema.

obtenerTitulo

Obtener un fichero digital que certifica el título obtenido por el ejemplar.

consultarTitulos

Obtiene la relación de títulos que un ejemplar ha obtenido hasta ese momento.



EXPOSICIONES Y CONCURSOS

Registra los resultados de los concursos y exposiciones de Morfología, Trabajo y disciplina, Rastreo y Agility que las diferentes federaciones, asociaciones y club realizan.

regarShow

Registrar los datos y características de una exposición o concurso (*entidad organizadora, tipo de concurso, ámbito, fecha, lugar, ...*)

regarResultadoShow

Registrar el resultado que un ejemplar ha obtenido al participar en una exposición o concurso, tanto a nivel de calificación como de puesto.

cagarResultadosShow

Realizar la carga masiva de todos los resultados de una exposición, con el fin de registrar las calificaciones y puestos obtenidos por cada ejemplar.

consultarResultadosShow

Obtiene información de los resultados y calificaciones de exposiciones y concursos.





El ámbito del proyecto puede expandirse a multitud de gestiones del mundo canino, que hoy día no han podido ser abordadas por el momento de manera global.

Por ejemplo:

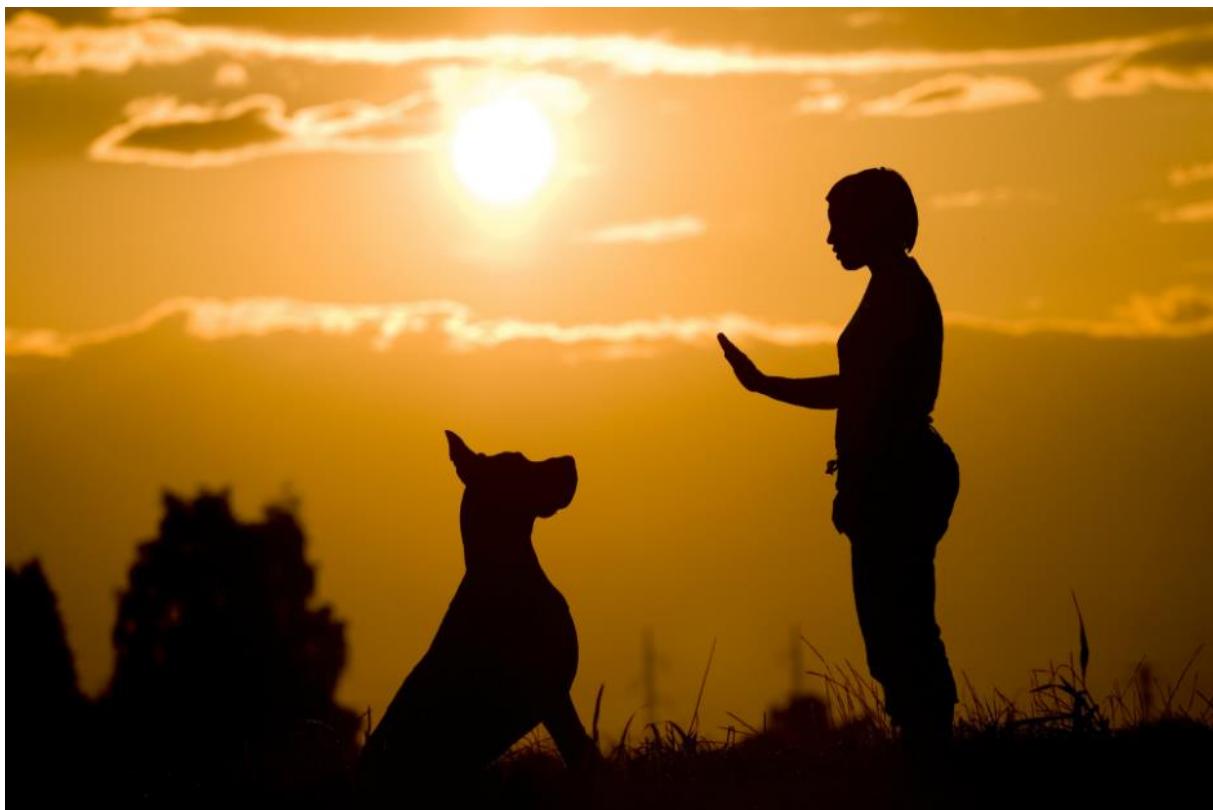
ADN

Contrato que gestiona el registro de ADN.

Facilitaría la certificación de autenticidad de la línea genealógica y permitiría el estudio genético a partir de los datos obtenidos.

ENFERMEDADES / TRATAMIENTOS

Contrato que gestiona el registro de las enfermedades y tratamientos que los veterinarios realizan sobre los perros, pudiéndose disponerse de un historial clínico para que cualquier veterinario pudiera consultarlos cuando llegue un ejemplar a su consulta.



Carga inicial de datos en la Blockchain

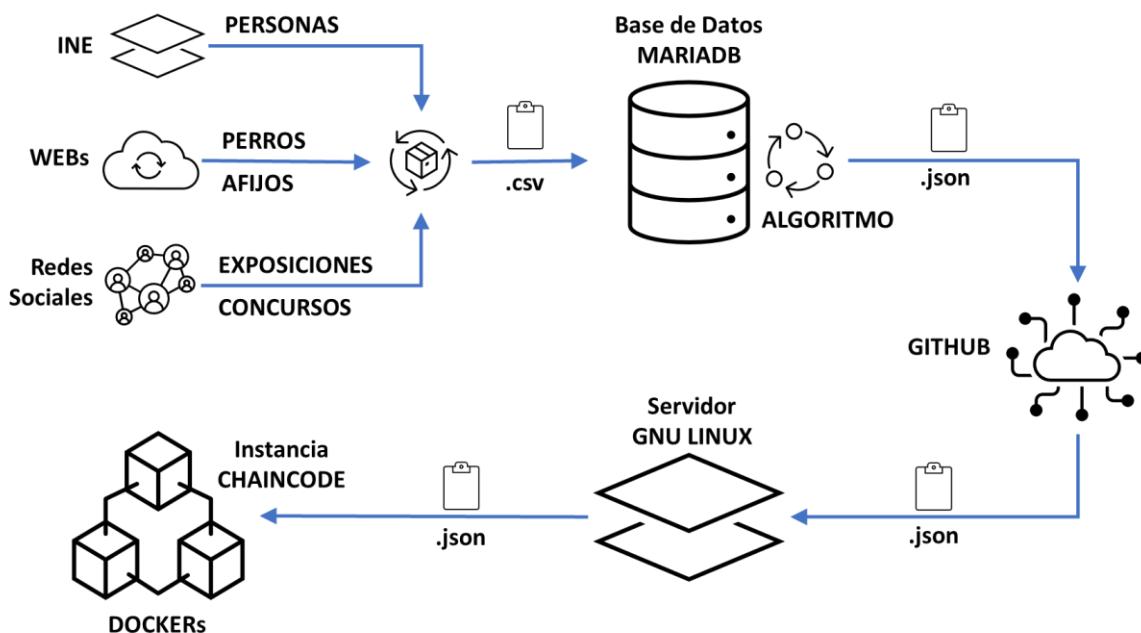
Introducción

A pesar de que hemos intentado que federaciones, asociaciones y clubes colaboren en el desarrollo del proyecto no hemos conseguido que nos proporcionen datos que sirvieran de base para cargar la cadena de bloques, más allá de lo que estas organizaciones presentaban en sus plataformas digitales, ya que consideran que los datos eran solo de uso interno.

Por tanto, ha sido necesaria la generación de una base de datos a partir de información que hemos podido extraer de organismos oficiales como el INE, y el depositado en páginas web y redes sociales.

Proceso de carga

El proceso de carga sigue el siguiente esquema:



A través de las consultas al INE se obtiene los 5000 apellidos más comunes en España, así como los 5000 nombres de mujeres y 5000 nombres de hombres.

A través de un proceso automático que los combina automáticamente, junto con un generador de NIF se obtiene la primera tabla con las PERSONAS que formarán parte de la organización en una Base de datos MariaDB.



En las pruebas iniciales conseguimos generar sin problema 100.000 usuarios, pero para la prueba de concepto reducimos a 1000 para no sobrecargar el tiempo de carga, aunque comprobamos que la blockchain lo podía soportar con los servidores que había cedido la universidad, siempre y cuando cargáramos los datos de 1000 en 1000.

ID	NOMBRE	APELLIDO_1	APELLIDO_2	SEXO	TIPO_DOCUMENTO	IDENTIFICADOR_DOCUMENTO	PAIS_EMISOR
1	CRISTINA	RODRIGUEZ	CHAMORRO	MUJER	NIF	0000010X	SPAIN
2	DANIEL	LANZAS	PELILCO	HOMBRE	NIF	93649739H	SPAIN
3	HELENA	GARCIA	FERNANDEZ	MUJER	NIF	01937444Q	SPAIN
4	JOSE	BENINANI	PAREJA	HOMBRE	NIF	00099322P	SPAIN
5	JUAN JOSE	LUCAS	DE LA FUENTE	HOMBRE	NIF	00012375R	SPAIN
6	UNAI	ARES	ICARAN	HOMBRE	NIF	30624115E	SPAIN
7	PILAR	SANTOS	PIENAS	MUJER	NIF	03080483R	SPAIN
8	SERGIO	TORRES	PALOMINO	HOMBRE	NIF	96669643X	SPAIN
9	JOSE CARLOS	SOTO	GOMEZ	HOMBRE	NIF	6244456P	SPAIN
10	ISABEL MARINA	CINTADO	VIDEZ	MUJER	NIF	33219695J	SPAIN
11	YOVANA	PEREZ	ALFARO	MUJER	NIF	36380101N	SPAIN
12	PEDRO ALEXIS	NUEZ	BAUZA	HOMBRE	NIF	10269993X	SPAIN
13	ANGELA PILAR	TARANCON	CORTADA	MUJER	NIF	32191194A	SPAIN
14	ANDRES RAMON	SORROCHE	FUENTES	HOMBRE	NIF	67541633V	SPAIN
15	DOMINGO ALBERTO	ARES	COLLANTES	HOMBRE	NIF	54121590Z	SPAIN
16	RAMI	VALDEPEÑAS	RODRIGUEZ	HOMBRE	NIF	70626408D	SPAIN
17	XENIX	FREIRE	PIENARANDA	HOMBRE	NIF	17290529A	SPAIN
18	SAVINIA	HERRADA	PELEGRI	MUJER	NIF	99663347T	SPAIN
19	BRIAN	DE LA LLAVE	OLIVERA	HOMBRE	NIF	46977749L	SPAIN
20	LAMBERTO	MASEGOSA	CARAZO	HOMBRE	NIF	15389455V	SPAIN
21	GENARO	ALEXANDRE	BAÑA	HOMBRE	NIF	33106304N	SPAIN
22	ANDREW	PAYAN	SALAS	HOMBRE	NIF	89108200I	SPAIN
23	SAAD	PACHO	ALCACER	HOMBRE	NIF	75972448K	SPAIN
24	ALMA MARIA	VEIGA	ZEA	MUJER	NIF	46414397Y	SPAIN
25	MARIA ALFONSA	MAÑAS	RUSO	MUJER	NIF	12477139F	SPAIN
26	MAR	LEMOS	GERMAN	MUJER	NIF	70362759D	SPAIN
27	VICENTE MACHA	TUR	AITA	NO ADOBE	NIF	77161046C	SPAIN

A través de algoritmos aleatorios se genera las restantes tablas:

- Los nombres de los Perros
- El afijo de criador al que pertenecen
- Los propietarios y los cambios de titularidad de los perros y de los afijos de criador
- La fecha de nacimiento y defunción
- La línea genealógica ascendente y descendente donde las fechas deben coincidir con la edad mínima y máxima de procreación de las progenitoras mientras estén vivos (la línea genealógica se puede configurar según se dese tomado como base para la prueba de concepto cargar a los perros con 20 generaciones anteriores)
- Los perfiles de los diferentes usuarios
- Los datos relativos a las colegiaturas de los veterinarios
- El historial de inserción de microchips y vacunación de los perros
- ...

The screenshot shows the HeidiSQL interface with the following details:

- Left Panel (Object List):** Shows the database structure under "CARGA_INICIAL_BlockChain". The "PERROS" table is selected, indicated by a grey background.
- Top Bar:** "Host: tfm.cnhygfwtvffy.eu-west-3.rds.amazonaws.com", "Base de datos: CARGA_INICIAL_BlockChain", "Procedimiento: _generar_TABLA_PERROS", "Consulta".
- Central Area (Procedure Definition):**
 - Opciones:** "Nombre: _generar_TABLA_PERROS", "Definidor: desarrollador@%", "Comentario: ", "Tipo: Procedimiento (no devuelve un resultado)", "Acceso a Datos: Contains SQL", "Bases de datos: Seguridad SQL: Definer", "Determinístico:
 - Cuerpo de la rutina:**

```
181    FOR numeroGeneracion IN 2..p_numeroGeneraciones
182    DO
183        SET done(FALSE);
184        OPEN cursor_afijos;
185        loop_afijos_generacion: LOOP
186            FETCH cursor_afijos INTO v_id_afijo;
187            IF done THEN
188                LEAVE loop_otras_generacion;
189            END IF;
190            OPEN cursor_hembras_afijo(v_id_afijo);
191            loop_hembras_afijo: LOOP
192                FETCH cursor_hembras_afijo INTO v_id_hembra, v_fecha_nacimiento, v_fecha_defuncion;
193                IF done THEN
194                    LEAVE loop_hembras_afijo;
195                END IF;
196                SET numero_perros = ( SELECT FLOOR(1 + (RAND() * 5)));
197                SET fecha_nacimiento_camada = ( SELECT CURDATE() + INTERVAL (2*(numeroGeneracion-1)) YEAR + INTERVAL ( 365 + RAND() ) DAY);
198                SET id_macho = ( SELECT ID
199                                FROM PERROS
200                                ORDER BY RAND()
201                                LIMIT numero_perros
202                            );
203            END LOOP;
204            CLOSE cursor_hembras_afijo;
205        END LOOP;
206        CLOSE cursor_afijos;
207    END DO;
208
```
- Bottom Right (Buttons):** "Ejecutar rutina(s)..." button.

Una vez generada las tablas en la base de datos **MariaDB** se exportan a unos ficheros JSON que serán almacenados en GitHub en la siguiente dirección:

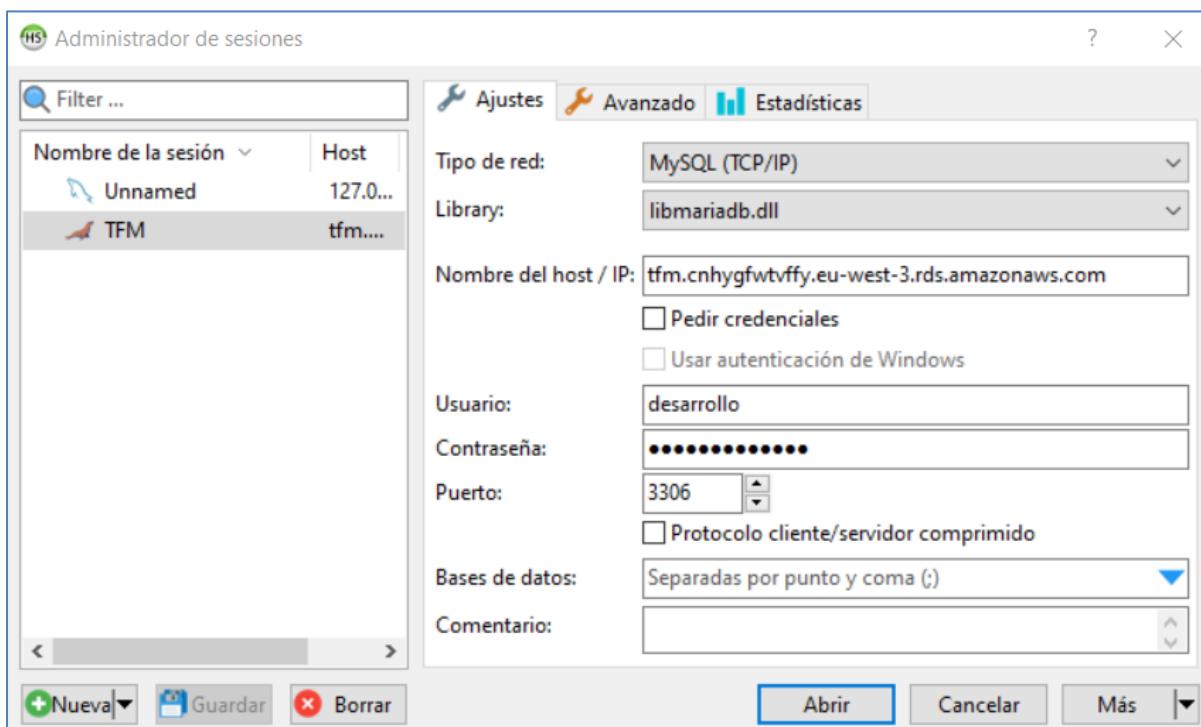
https://github.com/DFLBB/TFM_archs/tree/master/json

Cuando se arranca la red de blockchain los ficheros contenidos en esta carpeta son copiado junto con el resto de los scripts y código fuente de los contratos inteligentes en el servidor GNU Linux.

Al instalar e instanciar los contratos se copia los ficheros JSON dentro del Docker que lo contiene para su ejecución a través de las funciones de carga.

Databases			
	Database name	Size	Operations
_users	2.3 KB	1	
netcanchannel_	18.2 KB	2	
netcanchannel_afijos	31.4 KB	106	
netcanchannel_lscc	6.0 KB	9	
netcanchannel_microchips	1.5 MB	4762	
netcanchannel_profiles	16.3 KB	51	
netcanchannel_perros	3.1 MB	9524	
netcanchannel_personas	410.3 KB	1003	
netcanchannel razas	103.9 KB	352	
netcanchannel_solicitudes	0.5 KB	1	
netcanchannel_vacunas	37.9 KB	104	
netcanchannel_veterinarios	8.5 KB	24	

Para acceder a la Base de Datos de MariaDB lo puede hacer con la siguiente configuración:



- HOST: tfm.cnhygfwtvffy.eu-west-3.rds.amazonaws.com
- USUARIO: desarrollo
- CLAVE: tfmDesarrollo
- BASE DATOS: CARGA_INICIAL_Blockchain

También puede revisar estos procesos a través del script de SQL definido en:

https://github.com/DFLBB/TFM_archs/tree/master/script_mariadb_carga_inicial



VERSION
FINAL

15-9-2020

Blockchain & BigData Canino

ANEXO IV: Api-Rest y Web



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE

Tecnologías utilizadas.....	2
MariaDB	2
Apache Tomcat	2
Java	3
AWS	3
Maven	4
HTML	4
Bootstrap	5
CSS	6
JavaScript y JQuery	6
Servidor.....	7
Representación de los diferentes recursos.....	8
Login	8
Vacuna	9
Chips	9
Perro	9
Afijos	10
Diseño y definición de URIs.....	10
Diseño de la base de datos.....	11
Ejemplo de llamada	12
Página web.....	14
Próximas mejoras.....	17
Actualización de JSON.....	17
Actualización de las URIs.....	18
Actualización de la base de datos	19
Bibliografía.....	20



Tecnologías utilizadas

MariaDB

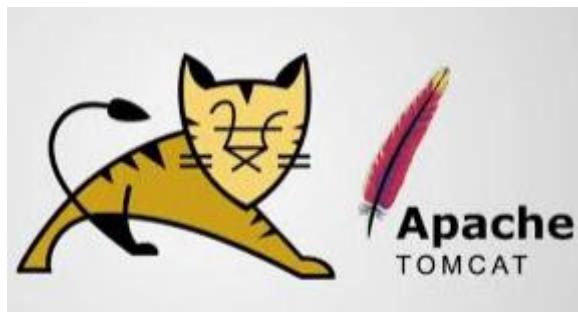
MariaDB es uno de los sistemas de gestión de bases de datos relacionales existentes actualmente. Fue lanzado el 29 de mayo de 2009 desarrollado por el fundador de MySQL Michale Widenius, la fundación MariDB y la comunidad de código abierto. [1]

La base de datos ha sido alojada en Amazon Web Services utilizando el servicio RDS con una versión de MariaDB 10.4.8.



Apache Tomcat

Apache Tomcat es un contenedor de Servlets desarrollado por la empresa Apache Software Foundation, cuyo lanzamiento fue el año 1999. Actualmente, su última versión data del 13 de mayo de 2019 cuyo número es la 9.0.20. Está desarrollado en el lenguaje de programación Java, es multiplataforma y funciona sobre una JVM (Java Virtual Machine). [2]



En cuanto a los diferentes entornos del proyecto, en el entorno de pruebas, el servicio REST fue probado en tres versiones diferentes de Tomcat, la 7, 8 y 8.5. La elección final fue escoger la versión 8.5, por lo que tenemos el servidor que gestiona el servicio en AWS con una versión de Apache Tomcat de 8.5, esta versión es la única que se nos ofrece en esta plataforma.



Java

Es un lenguaje de programación orientado a objetos que desarrollado con la intención de permitir a los desarrolladores escribir una vez el código del programa y que lo ejecuten en cualquier dispositivo. [3]

Su desarrollo empezó por James Gosling que trabajaba en la empresa Sun Microsystems, en 1996 fue publicado el primer JDK 1.0 (23 de enero de 1996). Si continuamos hasta actualidad, podemos avanzar hasta Java SE 14. A lo largo de este camino Java ha ido evolucionando en casi todos los aspectos, como por ejemplo, la incorporación de los interfaces de Runnable y Callable haciendo más fácil la programación concurrente en el lenguaje, aportando soporte para las conexiones a bases de datos con JDBC (Java Database Connectivity) en el JDK 1.1... [3]

Además, Java cuenta con un recolector de basura, es decir, Java runtime es el encargado de gestionar el ciclo de vida de los diferentes objetos que se van creando en la ejecución de un programa. [3] En este proyecto, Java es el encargado de la programación del servicio REST completo, es decir, todo el servicio está programado en Java 8.0, gracias a las diferentes librerías utilizadas como, JAX-RS una librería para el soporte de servicios web o el conector con las bases de datos MySQL mysql-connector-java. [4]



AWS

Amazon Web Services es una plataforma de computación en la nube creada por Amazon en el año 2006. Esta plataforma nos ofrece un total de más de 160 servicios en ella, entre ellas podemos destacar, el servicio de informática que nos permite lanzar máquinas, generar servicios web, el más importante de estos servicios es el EC2. El servicio de almacenamiento nos permite almacenar diferentes tipos de archivos en la nube, cabe destacar el servicio S3. Servicios de machine learning, análisis, bases de datos, robótica, seguridad, servicios multimedia... [5]

Actualmente, Amazon Web Services cuenta con 21 regiones como París, Tokio, Ohio, Canadá, etc. Además, de tener tres de ellas en proceso de lanzamiento en España, Osaka y Yakarta. [5]





Por otro lado, es la plataforma líder en servicios en la web por delante de Google, Microsoft, IBM. También, cuenta con grandes empresas como clientes como la Fórmula 1, Netflix, Vodafone o Adobe entre muchas otras. [5]

Para nuestro proyecto, AWS ha sido utilizado en el entorno de producción dando uso a los servicios EC2 para lanzar una máquina en la nube, el servicio Elastic beanstalk para lanzar un servidor Apache Tomcat y el servicio RDS para crear una base de datos MariaDB.

Maven

Maven ha sido el software de gestión de paquetes utilizado en Eclipse para desarrollar el servicio REST. Fue desarrollado en el año 2002 por Jason Van Zyl. Actualmente, es un proyecto de Apache Software Foundation. [6]

Este utiliza un archivo llamado POM (Project Object Model) que es el encargado de gestionar las dependencias de los diferentes módulos y encargada de construir la aplicación. Gracias a este archivo podemos añadir todas las librerías que necesitemos en un instante, solo deberemos añadir la etiqueta <dependencies> y dentro de ella añadir las dependencias que queramos añadir con el siguiente formato:

```
<dependency>
    <groupId> </groupId>
    <artifactId> </artifactId>
    <version></version>
</dependency>
```

Una vez añadido, Maven se encargará de descargar las librerías necesarias y añadirlas al proyecto. Además, cuenta con su propia página web donde encontrar las dependencias rápidamente (<https://mvnrepository.com/>).

HTML

HTML (HyperText Markup Language) es un lenguaje de programación orientado a la creación de páginas web. Es un estándar del consorcio World Wide Web (WWW). [7]

Este fue lanzando en el año 1993, sin embargo, su desarrollo comenzó en el año 1991 donde Tim Berners-Lee describió casi 20 elementos en el diseño inicial de HTML, actualmente 13 de estos casi 20 elementos que describió





siguen estando en el lenguaje. [7]

HTML es un lenguaje formado por etiquetas donde tenemos etiquetas de apertura `<p>` y etiquetas que nos indican el cierre de dicha etiqueta `</p>`. Actualmente, se encuentra en la versión conocida como HTML5 lanzada el 28 de octubre de 2014. [7]

En HTML5 encontramos una gran variedad de etiquetas para la definición de la página web divididas en diferentes secciones las cuales son, elemento raíz, metadatos del documento, scripting, secciones, agrupaciones de contenido, semántica a nivel de texto, ediciones, contenido incrustado, datos tabulares, formularios y elementos interactivos.

Las dos etiquetas más importantes y que no deben faltar en ninguna página web son las etiquetas: `<!doctype html>` y `<html>`. La primera define que el documento está bajo el estándar de HTML5 y la segunda representa el elemento raíz de un documento HTML. Todos los elementos que compondrán la página web deben estar dentro de las etiquetas de apertura y de cierre de `<html>`.

Bootstrap

Bootstrap es una librería de código abierto desarrollada por la empresa Twitter cuyo objetivo es proveer diseños con diferentes características a diferentes componentes de HTML para la creación de páginas web. Esto es posible gracias a la ayuda de los lenguajes de programación web CSS y JavaScript. [28]

Esta herramienta fue lanzada el 19 de agosto de 2011 y actualmente se encuentra en la versión 4.2.1 que fue lanzada el 21 de diciembre de 2018. [8]

Además del propio Twitter, este framework es utilizado por grandes empresas como la NASA o MSNBC.

Para poder utilizar este framework, solo necesitaremos añadir tres etiquetas script y una etiqueta link a nuestra página web HTML, se indican a continuación: [9]

The screenshot shows a website layout with a green header bar containing the text "Últimos artículos". Below the header, there is a grid of small dog icons. To the right of the grid are two green buttons labeled "REGISTRO" and "INICIO". At the bottom of the page, there is a footer section with the text "Razas Potencialmente Peligrosas (PPP)" and "Hace 5 minutos [Lee más](#)". Below this, there is a code block for a `<link>` tag:

```
<link rel="stylesheet"
      href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"
      integrity="sha384-gg0yR0iXCBMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T"
      crossorigin="anonymous">
```



```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965Dz00rT7abK41JStQIAqVgRVzbzo5smXKp4YfRvH+8abtTE1Pi6jizo"
crossorigin="anonymous">

</script> <script
src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js"
integrity="sha384-
U02eT0CpHqdSJQ6hJty5KVphtPhzWj9W01c1HTMGa3JDZwrnQq4sF86dIHNDz0W1"
crossorigin="anonymous">

</script> <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"
integrity="sha384-
JjSmVgyd0p3pXB1rRibZUAYoIIy60rQ6VrjIEaFF/nJGzIxFDsf4x0xIM+B07jRM"
crossorigin="anonymous"></script>
```

CSS

CSS o Hojas de estilo en cascada (Cascading Style Sheets en inglés) es un lenguaje que nos permite definir y añadir un diseño gráfico a diferentes lenguajes, en nuestro caso, al lenguaje HTML. [10]

Lanzado el 17 de diciembre de 1996 por la empresa CSS Working Group y actualmente en la versión CSS3. Junto a HTML y JavaScript son los lenguajes más utilizados a la hora de crear páginas webs.[10]

CSS es una herramienta muy potente que nos permite crear una gran variedad de estilos e incluso animaciones solo con su propio código.

JavaScript y JQuery

JavaScript apareció en el año 1995 desarrollado por Brendan Eich. Este es un lenguaje interpretado y utilizado en el lado cliente. Además, nos permite interactuar con el DOM (Document Object Model), es decir, nos permite interactuar con los diferentes elementos definidos en el archivo HTML de la página web. [11]

En cambio, en este proyecto JavaScript toma un plano secundario y la librería JQuery toma el papel principal.

JQuery es una librería de JavaScript creada por John Resig y que fue lanzada en enero de 2006. Esta nos permite interactuar con los elementos de los archivos HTML de una manera más sencilla y cómoda para el desarrollador que el propio lenguaje JavaScript. [12]

Para poder utilizar JQuery necesitaremos añadir la siguiente línea a nuestro archivo HTML con la definición de la página web:



```
<script src="https://code.jquery.com/jquery-3.4.1.js" integrity="sha256-WpOohJOqMqqyKL9FccASB900KwACQJpFTUBLTYOVvVU=" crossorigin="anonymous"></script>
```

En nuestro caso, la versión utilizada ha sido la 3.4.1, una de las últimas versiones lanzadas de la librería.

Para entender el acceso más sencillo a los elementos del DOM que nos permite esta librería vamos a ver algunos ejemplos.

Si queremos acceder a un elemento del DOM con un id establecido, en JavaScript necesitaremos utilizar la siguiente línea: `document.getElementById('valorId');`, en cambio con Jquery solo necesitaremos añadir la siguiente línea: `$(“#id”)`.

En el caso de querer acceder a un elemento con el atributo class, en JQuery solo utilizaremos `$(“.class”)`, por el otro lado, para JavaScript necesitaríamos utilizar, `document.getElementsByClassName(“class”);`

Pero la función más importante proporcionada por JQuery, es `$(document).ready(function(){})`, esta identifica si la página está preparada para interaccionar con el DOM.

Por último, JQuery contiene AJAX (Asynchronous JavaScript and XML) que nos permite realizar llamadas a servicios REST.

Servidor

Para una mayor disponibilidad del servicio, este servicio se ha introducido en Amazon Web Services (AWS), utilizando el servicio Elastic Beanstalk.

AWS nos permite elegir diferentes regiones alrededor de la Tierra, entre los que se encuentran diferentes países de Asia, diferentes estados de Estados Unidos o diferentes países de Europa. En nuestro caso, hemos elegido la región de París para obtener un mejor rendimiento en el apartado de latencia de la red.

Al utilizar el servicio Elastic Beanstalk, este asocia una instancia de EC2 donde se ejecutará nuestro servicio REST. Además, nos permite elegir varios tipos de servidores distintos para diferentes lenguajes de programación. En nuestro caso, hemos elegido la plataforma Tomcat 8.5 with Java 8, debido a que hemos desarrollado este sistema en Java.

Netcan-env-2

i-020ffaff1f49c43c7

En ejecución

t2.micro



Representación de los diferentes recursos

Para la representación de los diferentes objetos se ha utilizado JSON por motivo de comodidad a la hora de programación.

Seguidamente se muestran los esquemas de todos los JSON definidos para trabajar en el servicio REST:

Login

Cuenta con 5 parámetros fijos para el registro del usuario en el sistema:

- Email
- Nickname
- Tipo
- Password
- Teléfono

```
{  
    "email": "",  
    "nickname": "",  
    "tipo": "",  
    "password": "",  
    "telefono": "",  
    "propietario": {},  
    "veterinario": {},  
    "federacion": {}  
}
```

Además de estos 5 parámetros, este JSON contiene también la información que vamos a introducir en el sistema según el tipo de usuario que se vaya a introducir. Contamos con 3 tipos de usuarios, veterinario (representado por una V), propietario (representado por una P) y federación (representado por una F).

El propietario está representado por el siguiente JSON:

```
"propietario": {  
    "documento": "",  
    "pais": "",  
    "tipoDoc": "",  
    "paisEmisorDoc": "",  
    "nombre": "",  
    "apellido1": "",  
    "apellido2": "",  
    "direccion": "",  
    "ciudad": ""  
}
```

El veterinario está representado por el siguiente JSON:



```
    "veterinario":{  
        "colegiado":,  
        "nombre": "",  
        "apellido1": "",  
        "apellido2": "",  
        "clinicaVeterinaria": "",  
        "direccion": "",  
        "pais": "",  
        "ciudad": "",  
        "fechaBaja": ""  
  
    }  
}
```

Por último, la federación está representada por el JSON:

```
    "federacion":{  
        "nombre": "",  
        "pais": ""  
    }
```

Vacuna

```
{  
    "idPerro":,  
    "idVeterinario":,  
    "codVacuna":,  
    "fechaBaja": "",  
    "idProteccion":,  
    "fechaBajaProteccion": ""  
}
```

Chips

```
{  
    "idVeterinario":,  
    "codMicrochip":,  
    "idPerro":  
}
```

Perro

```
{  
    "nombre": "",  
    "sexo":,  
    "idPadre":,  
    "idMadre":,  
    "fechaNacimiento": ""  
}
```



Afijos

```
{  
    "nombre": "",  
    "idPersonal1": "",  
    "idPersona2": "",  
    "idPersona3": ""  
}
```

Diseño y definición de URIs

La URI principal comenzará por el nombre principal del servicio, en este caso, se llamará como el nombre que hemos establecido al sistema, “/netcan”, seguido de “/api” para identificar la API con los diferentes métodos. Para una mayor escalabilidad del sistema, la URL principal a las funciones del sistema termina con “/v1”, ya que nos encontramos en la versión 1 del sistema REST. Por todo esto, la URI principal del sistema REST es el siguiente:

- /netcan/api/v1/afijos
- /netcan/api/v1/afijos/baja
- /netcan/api/v1/bigdata/image
- /netcan/api/v1/chips
- /netcan/api/v1/login
- /netcan/api/v1/perros
- /netcan/api/v1/perros/baja
- /netcan/api/v1/vacunas

Por el otro lado, tenemos las URIs definidas para el api de Big Data:

- /bigData/api/v1/modelos/predicion

Diseño de la base de datos

En la versión actual del sistema, la base de datos MariaDB es utilizada para la gestión del acceso al sistema, es decir, su único objetivo es el manejo del registro y el inicio de sesión en la aplicación.

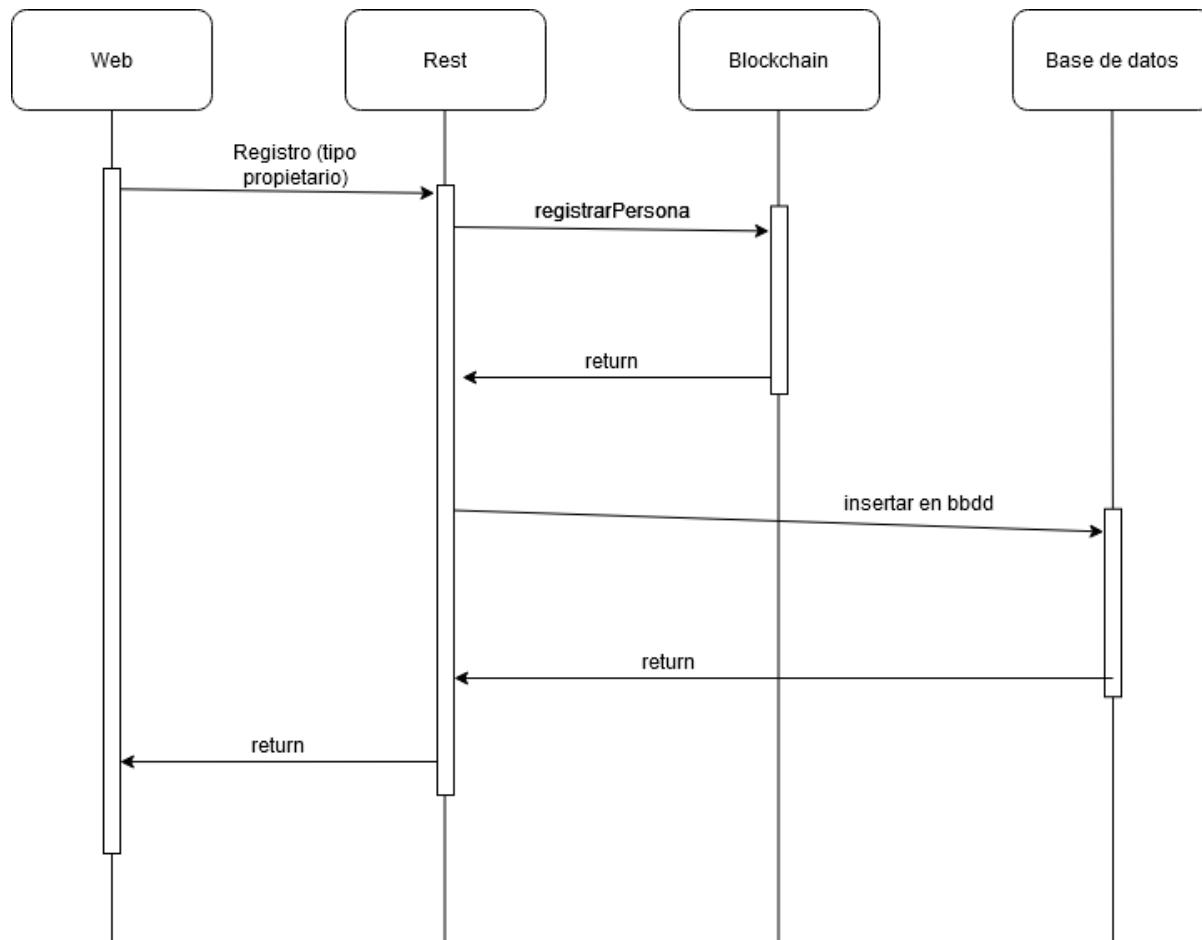
Para ello, se define una tabla “altas” que contiene varios campos obligatorios para el registro de los diferentes tipos de usuarios al sistema. Una vez registrado en la tabla “altas”, según el tipo de usuarios guardamos los datos correspondientes:



Ejemplo de llamada

En este apartado se mostrará dos ejemplos que muestran como interactúa los diferentes servicios con el servidor.

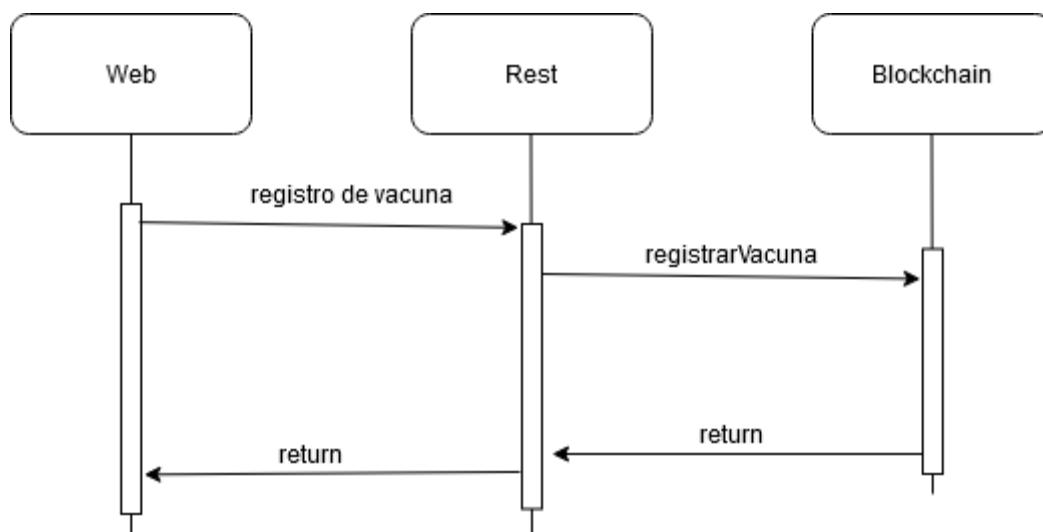
El primer caso se mostrará el sistema de alta en el sistema:



Observamos como el usuario hace el registro desde la página web contactando con el servicio rest y este es el encargado de insertarlo en la BlockChain y seguidamente, si todo ha ido correctamente en la BlockChain, se inserta en la base de datos.

Una vez insertado en los dos sistemas, el registro del usuario ha terminado exitosamente.

Por otro lado, tenemos las llamadas que solo atacan a la red BlockChain:

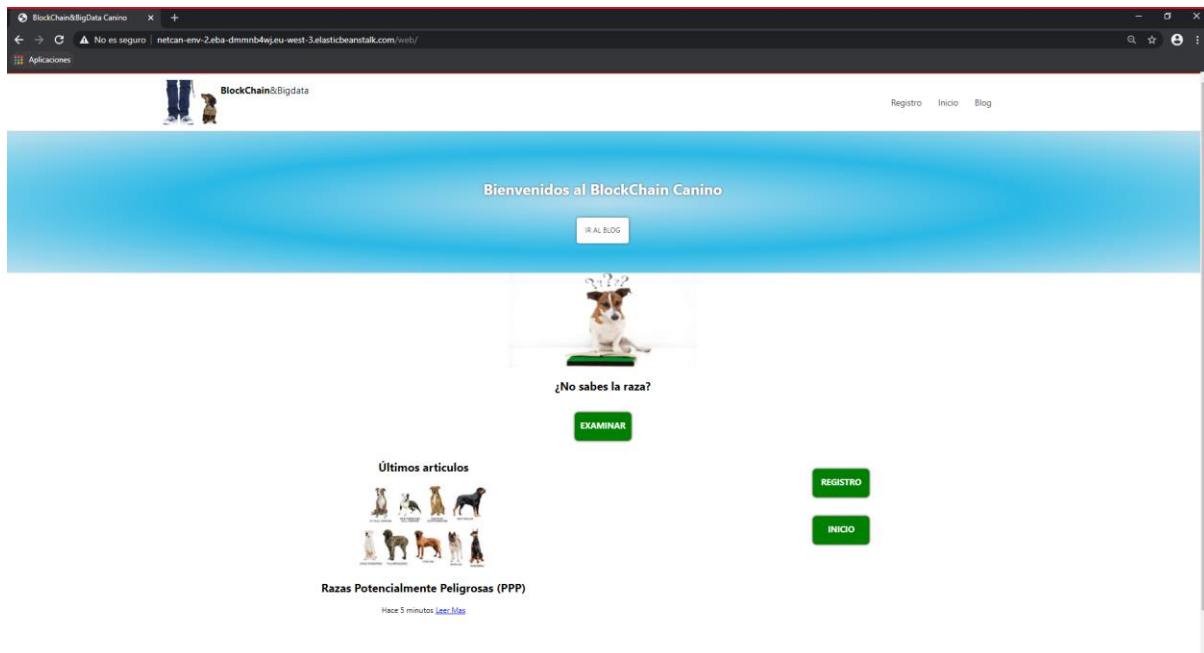


Página web

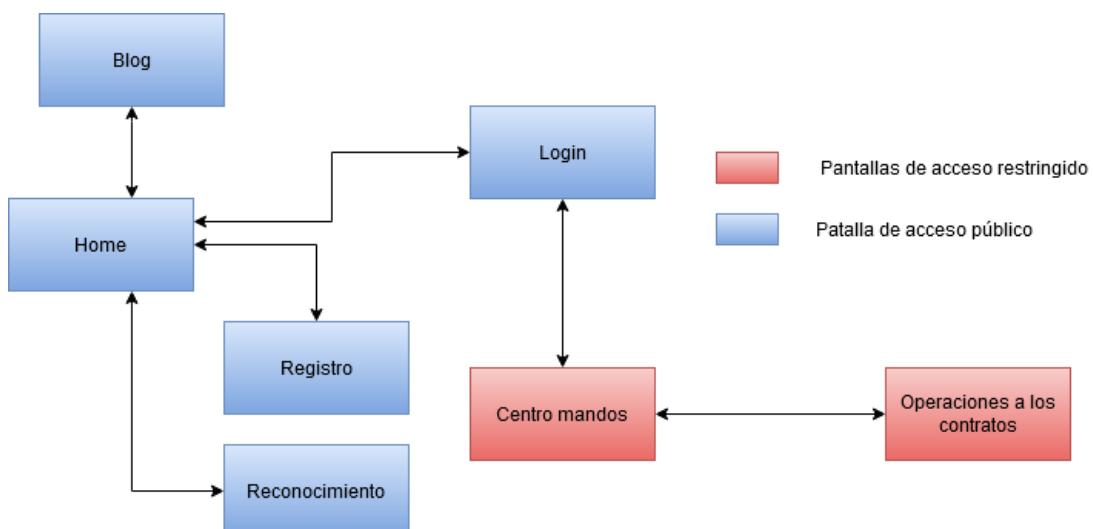
El portal web del aplicativo está alojado en el servidor de rest de entrada al sistema. Para acceder a dicho portal, utilizaremos la siguiente URL:

<http://netcan-env-2.eba-dmmnb4wj.eu-west-3.elasticbeanstalk.com/web/>

Una vez dentro encontraremos la página Home del portal web que tendrá la siguiente apariencia:



Desde la página Home, tendremos acceso a diferentes servicios dependiendo de si contamos con una cuenta o no en el aplicativo. Como se muestra en siguiente estructura, todo lo relativo a operaciones sobre BlockChain quedará restringido a navegaciones con usuario en el aplicativo. Por el contrario, acciones como el Blog o el Reconocimiento (reconocimiento de raza según una imagen) quedarán abiertos al público general:





Como se indicó anteriormente en el presente documento, contamos con 3 tipos de usuarios. Estos tres tipos contarán con diferentes opciones de operaciones sobre la BlockChain en la pantalla Centro de mandos.

En el caso de ser un usuario de tipo veterinario, encontraremos la siguiente apariencia en nuestro Centro de mandos:

view-source:netcan-env-2.eba-dmmb4wj.eu | netcan-env-2.eba-dmmb4wj.eu | BlockChain&BigData Canino | +
netcan-env-2.eba-dmmb4wj.eu-west-3.elasticbeanstalk.com/web/login/mainWeb/index.html?id=5&tipo=V

BlockChain&Bigdata

Home Blog

Cuadro de mandos

Registrar un perro
Aquí podrás registrar un perro nuevo en el sistema

Registrar defunción de un perro
Aquí podrás registrar la defunción de tu amigo/a

Registrar un chip
Aquí podrás registrar un chip nuevo en el sistema

Registrar vacuna de un perro
Aquí podrás registrar la vacuna en el sistema

Registrar Registrar Registrar Registrar

Se observa como este tipo de usuarios cuentan con 4 operaciones, registrar un perro, registrar la defunción de un perro, registrar el chip y registrar una vacuna de un perro. Sin embargo, para un perfil de tipo federación, el Cuadro de mandos tiene la siguiente apariencia:

BlockChain&Bigdata

Home Blog

Cuadro de mandos

Registrar un afijo
Aquí podrás registrar un afijo nuevo en el sistema

Registrar baja de un afijo
Aquí podrás registrar la baja de un afijo

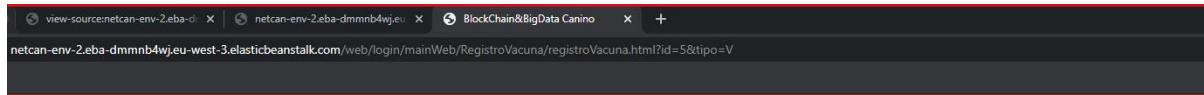
Registrar Registrar



Las federaciones cuentan con dos operaciones relacionadas con los afijos, registrar afijo y registrar baja de un afijo.

Para el caso de un usuario de tipo propietario, tendremos las operaciones registrar perro y registrar defunción de un perro.

Al seleccionar cualquiera de las operaciones que se han indicado anteriormente, accederemos a diferentes formularios que hablarán con el BlockChain. Por ejemplo, se muestra el registro de una vacuna:



BlockChain&Bigdata

Cuadro de mandos

Registro Vacunas

Formulario Alta

Identificador Perro

Identificador Veterinario

Codigo Vacuna

Fecha de Administracion

Duracion Dosis

Descripcion Vacuna

Enviar



Próximas mejoras

Actualización de JSON

Se añadirán nuevos JSON para una mejor experiencia de usuario de la página web para todos los tipos de usuarios.

En el caso de un usuario de tipo veterinario, la página web tendrá la información del veterinario, los chips que ha insertado y las vacunaciones realizadas por dicho veterinario. Para ello, se define el siguiente JSON:

Para el tipo de usuarios de federación, se define un JSON con la información de dicha federación y los concursos que organizan:

```
{
  "federacion": {
    "id": ,
    "nombre": ,
    "pais": "",
    "idAlta": ,
  },
  "concursos": {
    "concurso": [
      {
        "idConcurso": ,
        "nombre": ,
        "patrocinadores": ,
        "pais": ""
      },
      ...
      {
        "idConcurso": ,
        "nombre": ,
        "patrocinadores": ,
        "pais": ""
      }
    ]
  }
}
```

```
{
  "veterinario": {
    "colegiado": ,
    "nombre": "",
    "apellido1": "",
    "apellido2": "",
    "clinicaVeterinaria": "",
    "direccion": "",
    "pais": "",
    "ciudad": "",
    "idAlta": ,
  },
  "listaChips": {
    "chip": [
      {
        "codigo": ,
        "marca": "",
        "colegiado": ,
        "pais": ,
      },
      ...
      {
        "codigo": ,
        "marca": ,
        "colegiado": ,
        "pais": ,
      }
    ]
  },
  "vacunaciones": {
    "vacunacion": [
      {
        "perroNombre": ,
        "afijo": ,
        "vacuna": ,
        "fecha": ""
      },
      ...
      {
        "perroNombre": ,
        "afijo": ,
        "vacuna": ,
        "fecha": ""
      }
    ]
  }
}
```

Finalmente, para el tipo de usuario propietario, se define un nuevo JSON con la información del propietario y toda la información de los perros de dicho propietario:



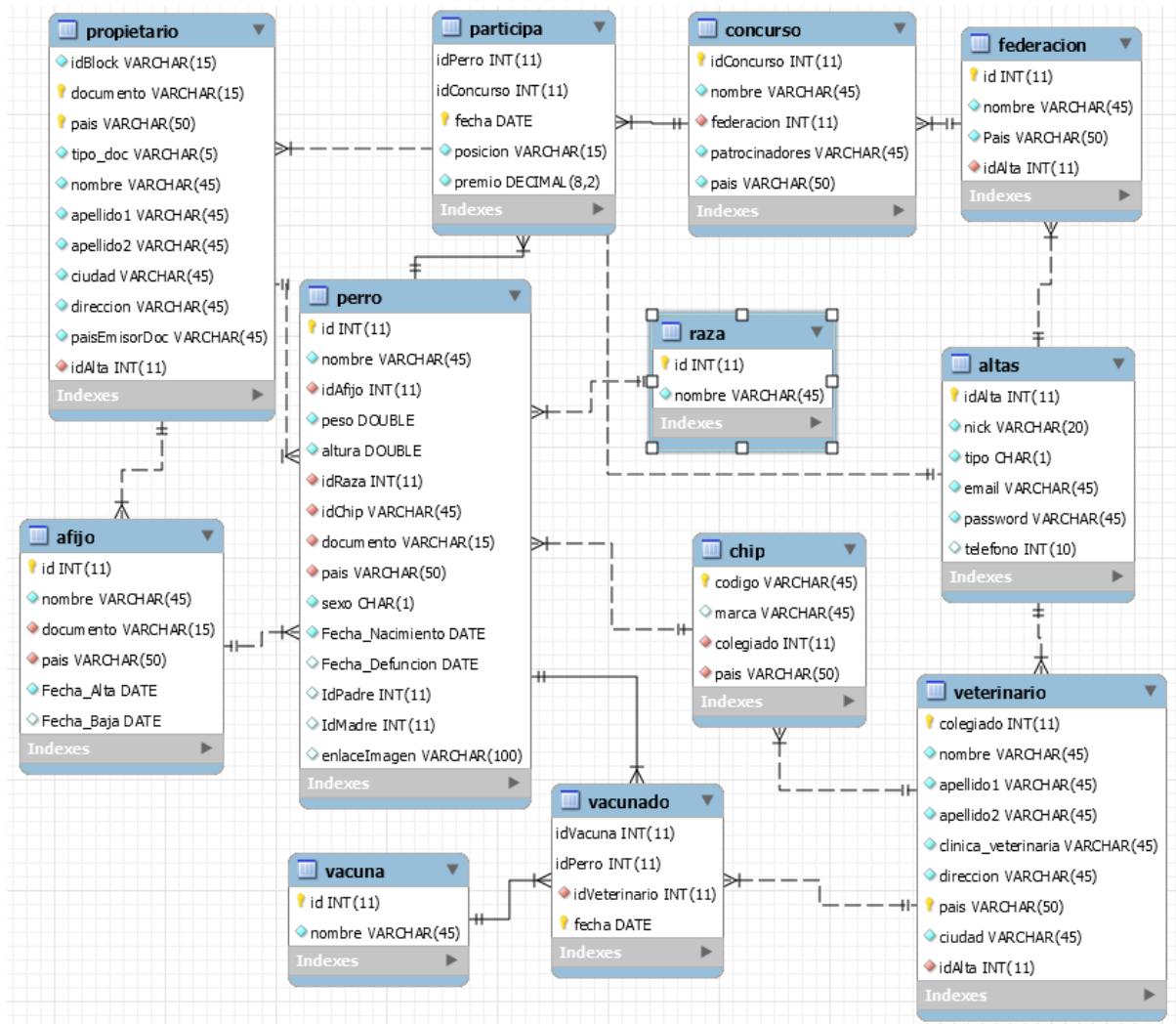
```
{
  "propietario": {
    "documento": "",
    "pais": "",
    "tipoDoc": "",
    "paisEmisorDoc": "",
    "nombre": "",
    "apellido1": "",
    "apellido2": "",
    "direccion": "",
    "ciudad": ""
  },
  "perros": {
    "perro": [
      {
        "nombre": ,
        "afijo": "",
        "peso": "",
        "altura": "",
        "raza": "",
        "sexo": ,
        "Fecha nacimiento": "",
        "Fecha defuncion": ,
        "nombre padre": ,
        "nombre madre": ,
        "enlace imagen": "
      },
      ...
      {
        "nombre": ,
        "afijo": "",
        "peso": "",
        "altura": ,
        "raza": ,
        "sexo": ,
        "Fecha nacimiento": ,
        "Fecha defuncion": ,
        "nombre padre": ,
        "nombre madre": ,
        "enlace imagen": "
      }
    ]
  }
}
```

Actualización de las URLs

Al añadir diferentes recursos para la actualización del servicio a la versión 2.0, se define la siguiente URI para mejorar la experiencia de usuario en el front con los nuevos JSON definidos:

- /netcan/api/v1/web

Actualización de la base de datos





Bibliografía

- [1] Wikipedia; MariaDB: <https://es.wikipedia.org/wiki/MariaDB>
- [2] Apache; Apache Tomcat: <http://tomcat.apache.org/>
- [3] Wikipedia; Java:
[https://es.wikipedia.org/wiki/Java_\(lenguaje_de_programaci%C3%B3n\)](https://es.wikipedia.org/wiki/Java_(lenguaje_de_programaci%C3%B3n))
- [4] Wikipedia; JAX-RX: <https://es.wikipedia.org/wiki/JAX-RS>
- [5] Amazon; ¿Qué es AWS?: <https://aws.amazon.com/es/what-is-aws/>
- [6] Wikipedia; Maven: <https://es.wikipedia.org/wiki/Maven>
- [7] Wikipedia; HTML: <https://es.wikipedia.org/wiki/HTML>
- [8] Wikipedia; Bootstrap:
[https://es.wikipedia.org/wiki/Bootstrap_\(framework\)#cite_note-3](https://es.wikipedia.org/wiki/Bootstrap_(framework)#cite_note-3)
- [9] Bootstrap; Introduction: <https://getbootstrap.com/docs/4.3/getting-started/introduction/>
- [10] Wikipedia; Hoja de estilos en cascada:
https://es.wikipedia.org/wiki/Hoja_de_estilos_en_cascada
- [11] Wikipedia; JavaScript: <https://es.wikipedia.org/wiki/JavaScript>
- [12] Wikipedia; JQuery: <https://es.wikipedia.org/wiki/JQuery>



Blockchain & BigData Canino

ANEXO V: Modelo de red neuronal convolucional para la identificación de razas de perro.



Autores:

Cristina Rodríguez Chamorro
Daniel Lanzas Pellico
Helena García Fernández
José Bennani Pareja
Juan José Lucas de la Fuente
Unai Ares Icaran

Tutor:

Sergio Torres Palomino

Documentación del TFM

Máster Blockchain y Big Data. Curso 2019-2020



ÍNDICE

Contenido

Introducción	2
MOTIVACIÓN	2
OBJETIVOS PRINCIPALES	2
• Objetivo general.....	2
• Objetivos específicos	2
ESTRUCTURA DEL DOCUMENTO Y TECNOLOGÍAS UTILIZADAS	3
• Estructura del documento	3
• Tecnologías utilizadas	3
Conceptos básicos sobre redes neuronales convolucionales (CNNs)	4
Análisis y diseño del modelo	7
ANÁLISIS Y DISEÑO DEL MODELO	7
CONFIGURACIÓN DEL SERVIDOR AMAZON LINUX (DEEP LEARNING VERSIÓN 33.0)	11
ESTUDIOS DE LOS MEJORES MODELOS CNNs	15
DEFINICIÓN Y ENTRENAMIENTO DE LOS MODELOS	16
• Definición de los modelos.....	16
• Preentrenamiento y entrenamiento de las capas añadidas.....	18
COMPARACIÓN DE LOS MODELOS Y SELECCIÓN DEL MEJOR	21
Generación de un Apache Tomcat en el servidor de Amazon Linux de Deep Learning.....	25
Predicciones con los datos de prueba	27
Conclusiones y líneas futuras.....	30
CONCLUSIONES	30
LÍNEAS FUTURAS	30
Bibliografía	31



Introducción

MOTIVACIÓN

Actualmente, conocemos el gran impacto y desarrollo que está teniendo el concepto del Deep Learning en el campo de la inteligencia artificial. Ya que, el claro avance de la tecnología está permitiendo que las empresas, industrias, e incluso universidades, despierten un notable interés por la obtención de nuevas técnicas que puedan aplicarse en su beneficio, para la generación de nuevas aplicaciones. (ARTEAGA, 2015). El objetivo del estudio del aprendizaje automático se basa en simular la inteligencia humana de forma artificial. De manera que, el Deep learning ha revolucionado el estado del arte, haciendo posible realizar tareas como el reconocimiento de voz, visión artificial etc. (Cárdenas, 2018).

Por todo ello, este trabajo se va a centrar en la aplicación de las técnicas predominantes del Deep Learning, en las que se hace uso de las redes neuronales convolucionales, para la clasificación de razas de perro. Ya que, se ha creído conveniente aprovechar esta herramienta tan interesante, para añadir funcionalidad a la parte Web del presente proyecto, y, de esta manera, hacer posible que cualquier usuario pueda reconocer la raza de un perro.

OBJETIVOS PRINCIPALES

- Objetivo general

Generar un modelo de Red Neuronal Convolutacional para la clasificación de razas de perro.

- Objetivos específicos

- Obtención de un dataset con más de 20000 imágenes de perros, y 120 razas distintas.
- Obtención del conjunto de datos de prueba y entrenamiento.
- Búsqueda de los mejores modelos preentrenados, y entrenamiento con distintos modelos.
- Comparación de los modelos entrenados y obtención del mejor modelo.
- Realización de las predicciones con los datos de prueba, y análisis de los resultados.
- Generar un script que permita la obtención de la predicción de la imagen que el usuario introduzca en la página web.

ESTRUCTURA DEL DOCUMENTO Y TECNOLOGÍAS UTILIZADAS

A continuación, se presenta la estructura de esta parte del proyecto, con el objetivo de comprender los pasos que se han seguido para implementar el modelo en cuestión.

- Estructura del documento

- **Introducción.** El capítulo 1 comprende la introducción y planteamiento al desarrollo del algoritmo para la clasificación de imágenes.
- **Conceptos básicos sobre redes neuronales convolucionales.** El capítulo 2 tiene como objetivo presentar los conceptos básicos de las redes neuronales convolucionales, para comprender su estructura y organización. Ya que, son las redes con las que se va a trabajar para conseguir los objetivos propuestos en el capítulo 1.
- **Análisis y diseño del modelo.** El capítulo 3 tiene como objetivo presentar todas las fases de análisis y diseño que se ha llevado a cabo para la implementación del algoritmo en cuestión.
- **Predicciones del conjunto de prueba.** El capítulo 4, tiene como objetivo mostrar el resultado de las predicciones realizadas sobre el conjunto de prueba.
- **Conclusiones y líneas futuras.** Este capítulo tiene como objetivo evaluar el cumplimiento de los objetivos iniciales, y la presentación de las líneas futuras que podrían complementar el trabajo realizado.

- Tecnologías utilizadas

- **Anaconda Navigator.** Es una GUI de escritorio que viene con Anaconda Individual Edition. Facilita el lanzamiento de aplicaciones y la administración de paquetes y entornos sin usar comandos de línea de comandos. Se ha utilizado para el uso de Python en el desarrollo del notebook para la implementación del modelo de red.
- **PuTTy.** Es un cliente SSH, Telnet, rlogin, y TCP raw con licencia libre. Se ha utilizado para la conexión con el servidor de Amazon Linux.
- **MobaXTerm.** Es una herramienta todo en uno que busca llevar a los usuarios más profesionales de Windows determinadas funciones de Linux como el uso de los comandos más habituales para controlar el sistema operativo desde el teclado. De manera que, se ha utilizado para las conexiones con el servidor de Amazon y para la cargar de datos. Ya que nos ofrece una consola mucho más visual.
- **Servidor Deep Learning Amazon Linux.** Se ha utilizado para el entrenamiento de los modelos, ya que, debido a la gran cantidad de datos utilizada, ha resultado imposible trabajar en local.



Conceptos básicos sobre redes neuronales convolucionales (CNNs)

La red neuronal convolucional es un tipo de red multicapa que presenta una alternancia entre las capas de convolución y de agrupación, finalizando con una red de neuronas artificiales convencional. (García, 2019).

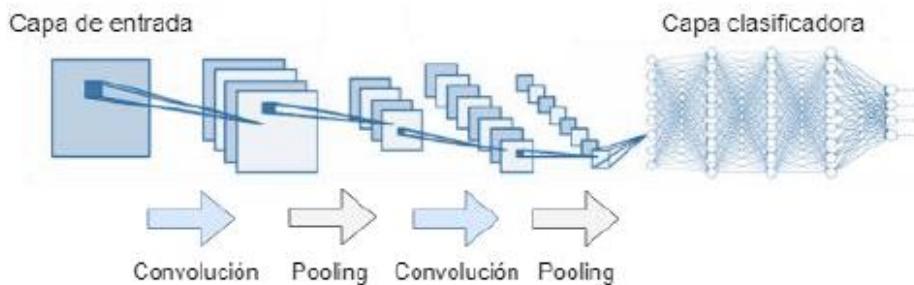


Figura 1. Arquitectura de una red CNN (García, 2019)

Capa de entrada: Capa destinada al procesamiento de los píxeles de la imagen introducida. Es decir, está formada por tantos píxeles como tenga la imagen. Es importante comentar que, aunque la imagen sea una matriz de píxeles, y el valor de los píxeles vaya de 0 a 255, para una red neuronal se normaliza de 0 a 1.

Capas ocultas: Capas destinadas a la obtención de las características más importantes de las imágenes. De manera que, las primeras capas son capaces de detectar líneas, curvas, y se van especializando hasta llegar a las capas más profundas. Dentro de estas capas, encontramos: (García, 2019).

Capas de convolución: Estas capas son las encargadas de aplicar los filtros necesarios para conseguir las características principales o los patrones más característicos de las imágenes recibidas. Por tanto, tras el preprocesamiento de la imagen, comienza el proceso distintivo de las CNN. De manera que, tienen lugar las llamadas convoluciones, en las que se toman grupos de píxeles de la imagen de entrada para realizar el producto escalar contra una pequeña matriz denominada kernel. Es decir, el kernel será el encargado de ir recorriendo todas las neuronas de entrada, dando como resultado una nueva capa de neuronas ocultas. (Pacheco, 2017). Por tanto, a medida que se va desplazando el kernel, se va obteniendo una nueva imagen filtrada por el kernel. De manera que, las imágenes nuevas dibujan las características de la imagen original. A continuación, se muestra una convolución con un cierto Kernel, para la comprensión de lo anteriormente comentado.

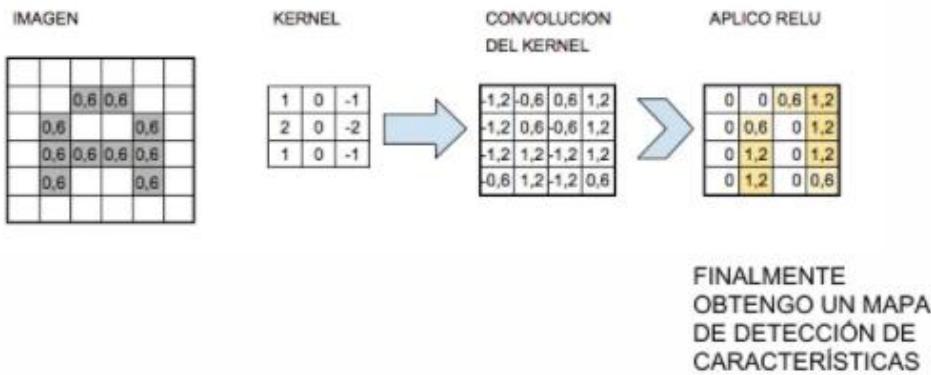


Figura 2. Ejemplo de una convolución con un kernel determinado. Nota: la función de activación consiste en $f(x)=\max(0,x)$. (Bagnato, 2018)

Capas de agrupación o pooling: Realizan la reducción del tamaño de la imagen, para poder trabajar con un menor número de píxeles y mejorar el rendimiento y la calidad del algoritmo. (Bagnato, 2018).

Posteriormente a la convolución, viene el paso del pooling o subsampling. Es decir, después de la convolución se produce una reducción de la cantidad de neuronas antes de realizar una nueva convolución. Esto es porque el número de neuronas después de la primera convolución genera una capa oculta del número de neuronas iniciales por el número de mapas que se incluyan en la convolución. Es decir, si tenemos una foto en blanco y negro 28x28px y 32 mapas de características, tendremos inicialmente 784 neuronas, y después de la convolución, 25088 neuronas. Lo cual quiere decir que, si se continúan realizando convoluciones, el número de neuronas resultantes en la segunda convolución, será demasiado grande, por lo que el procesamiento necesario para esta red sería muy elevado. (Bagnato, 2018). Por esta razón aparece el concepto de pooling o subsampling. Ya que, gracias a este procedimiento se va a reducir considerablemente el tamaño de las imágenes filtradas. De manera que, se obtendrán las características más importantes de cada una de ellas, para reducir el tamaño de las neuronas antes de una nueva convolución. Es importante comentar que, encontramos 2 tipos de pooling:

- Max pooling: Procedimiento a partir del cual se selecciona el mayor valor de los píxeles que intervienen. A continuación, se puede observar un ejemplo de max pooling:

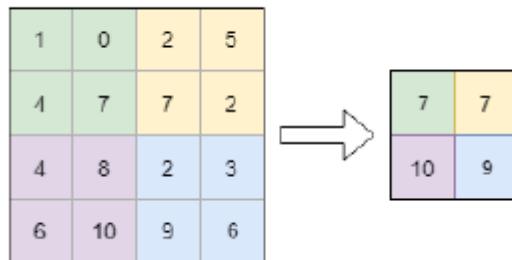


Figura 3. Ejemplo max pooling (García, 2019)

- Average pooling: Este procedimiento consiste en calcular la media de los píxeles de la matriz filtrada. A continuación, se puede observar en la figura 6 un ejemplo de average pooling.

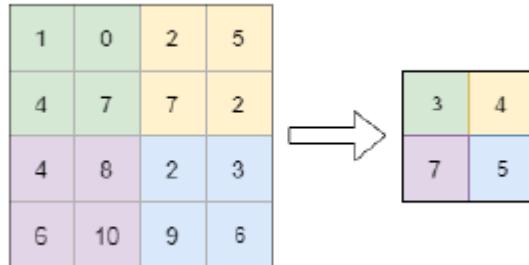


Figura 4. Ejemplo de Average Pooling. (García, 2019)

Red de neuronas artificial convencional: Red de neuronas que conectará con la última capa de subsampling y finalizará con la cantidad de neuronas que queremos clasificar. Utilizará backpropagation para ajustar los pesos de todas las interconexiones de las capas. Sin embargo, en la red CNN también se utilizará este procedimiento para ajustar los pesos de los kernels.



Análisis y diseño del modelo

ANÁLISIS Y DISEÑO DEL MODELO

Para la obtención de un mejor modelo de red neuronal convolucional que sea capaz de identificar razas de perros, ha sido necesario hacer uso de las 120 razas de perro utilizadas en el dataset Standford Dog DataSet. Este conjunto de datos se ha creado utilizando imágenes y anotaciones de ImageNet para la tarea de categorización detallada de imágenes. (Li, s.f.) . Dicho Dataset es accesible desde el siguiente sitio web: <https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>.

De manera que, se han utilizado 120 categorías de perro y 22.580 imágenes, distribuidas en 9960 imágenes para el conjunto de prueba y 10620 para el conjunto de entrenamiento. Es importante comentar que, para la obtención del conjunto de entrenamiento y del conjunto de prueba, se ha llevado a cabo la realización de un script que añade 83 imágenes de cada categoría a la carpeta Images_test. El path de dicha carpeta se utilizará posteriormente para realizar las predicciones sobre el conjunto de prueba, es decir, sobre imágenes de distintas razas de perro que no se van a utilizar para entrenar los modelos.

Además de todo ello, para lograr un óptimo análisis exploratorio de los datos, se ha desarrollado en el notebook ModeloBigData-TFM.ipynb realizado, un código que te permite conocer el nombre de las razas de perro utilizadas, y el número de imágenes correspondientes a cada una de ellas, para el conjunto de entrenamiento. El resultado referente a la implementación de dicho código se puede apreciar a continuación:

```
INFO:root: Directorio del dataset de imágenes: /home/ec2-user/images/Images
INFO:root: Buscando la 7ctivación7 del dataset
INFO:root: Numero de clases: 120 con nombres: ['n02085620-Chihuahua', 'n02085782-Japanese_sparniel', 'n02085936-Maltese_dog', 'n02086079-Pekinese', 'n02086240-Shih-Tzu', 'n02086646-Blenheim_spaniel', 'n02086910-papillon', 'n02087046-toy_terrier', 'n02087394-Rhodesian_ridgeback', 'n02088094-Afghan_hound', 'n02088238-basset', 'n02088364-beagle', 'n02088466-bloodhound', 'n02088632-bluetick', 'n02089078-black-and-tan_coonhound', 'n02089867-Walker_hound', 'n02089973-English_foxhound', 'n02090379-redbone', 'n02090622-borzoi', 'n02090721-Irish_wolfhound', 'n02091032-Italian_greyhound', 'n02091134-whippet', 'n02091244-Ibizan_hound', 'n02091467-Norwegian_elkhound', 'n02091635-otterhound', 'n02091831-Saluki', 'n02092002-Scottish_deerhound', 'n02092339-Weimaraner', 'n02093256-Staffordshire_bullterrier', 'n02093428-American_Staffordshire_terrier', 'n02093647-Bedlington_terrier', 'n02093754-Border_terrier', 'n02093859-Kerry_blue_terrier', 'n02093991-Irish_terrier', 'n02094114-Norfolk_terrier', 'n02094258-Norwich_terrier', 'n02094433-Yorkshire_terrier', 'n02095314-wire-haired_fox_terrier', 'n02095570-Lakeland_terrier', 'n02095889-Sealyham_terrier', 'n02096051-Airedale', 'n02096177-cairn', 'n02096294-Australian_terrier', 'n02096437-Dandie_Dinmont', 'n02096585-Boston_bull', 'n02097047-miniature_schnauzer', 'n02097130-giant_schnauzer', 'n02097209-standard_schnauzer', 'n02097298-Scotch_terrier', 'n02097474-Tibetan_terrier', 'n02097658-silky_terrier', 'n02098105-soft-coated_wheaten_terrier', 'n02098286-West_Highland_white_terrier', 'n02098413-Lhasa', 'n02099267-flat-coated_retriever', 'n02099429-curly-coated_retriever', 'n02099601-golden_retriever', 'n02099712-Labrador_retriever', 'n02099849-Chesapeake_Bay_retriever', 'n02100236-German_short-haired_pointer', 'n02100583-vizsla', 'n02100735-English_setter', 'n02100877-Irish_setter', 'n02101006-Gordon_setter', 'n02101388-Brittany_spaniel', 'n02101556-clumber', '
```



*n02102040-English_springer', 'n02102177-Welsh_springer_spaniel', 'n02102318-cocker_spaniel', 'n02102480-Sussex_spaniel', 'n02102973-Irish_water_spaniel', 'n02104029-kuvasz', 'n02104365-schipp
erke', 'n02105056-groenendael', 'n02105162-malinois', 'n02105251-briard', 'n02105412-kelpie', 'n02
105505-komondor', 'n02105641-Old_English_sheepdog', 'n02105855-Shetland_sheepdog', 'n021060
30-collie', 'n02106166-Border_collie', 'n02106382-Bouvier_des_Flandres', 'n02106550-Rottweiler', 'n
02106662-German_shepherd', 'n02107142-Doberman', 'n02107312-miniature_pinscher', 'n0210757
4-Greater_Swiss_Mountain_dog', 'n02107683-Bernese_mountain_dog', 'n02107908-Appenzeller', 'n
02108000-EntleBucher', 'n02108089-boxer', 'n02108422-bull_mastiff', 'n02108551-Tibetan_mastiff',
'n02108915-French_bulldog', 'n02109047-Great_Dane', 'n02109525-Saint_Bernard', 'n02109961-Esk
imo_dog', 'n02110063-malamute', 'n02110185-Siberian_husky', 'n02110627-affenpinscher', 'n02110
806-basenji', 'n02110958-pug', 'n02111129-Leonberg', 'n02111277-Newfoundland', 'n02111500-Gre
at_Pyrenees', 'n02111889-Samoyed', 'n02112018-Pomeranian', 'n02112137-chow', 'n02112350-kees
hond', 'n02112706-Brabancon_griffon', 'n02113023-Pembroke', 'n02113186-Cardigan', 'n02113624-
toy_poodle', 'n02113712-miniature_poodle', 'n02113799-standard_poodle', 'n02113978-Mexican_h
airless', 'n02115641-dingo', 'n02115913-dhole', 'n02116738-African_hunting_dog']*

INFO:root: Numero de imágenes de la clase n02085620-Chihuahua: 69

INFO:root: Numero de imágenes de la clase n02085782-Japanese_spaniel: 102

INFO:root: Numero de imágenes de la clase n02085936-Maltese_dog: 169

INFO:root: Numero de imágenes de la clase n02086079-Pekinese: 66

INFO:root: Numero de imágenes de la clase n02086240-Shih-Tzu: 131

INFO:root: Numero de imágenes de la clase n02086646-Blenheim_spaniel: 105

INFO:root: Numero de imágenes de la clase n02086910-papillon: 113

INFO:root: Numero de imágenes de la clase n02087046-toy_terrier: 89

INFO:root: Numero de imágenes de la clase n02087394-Rhodesian_ridgeback: 89

INFO:root: Numero de imágenes de la clase n02088094-Afghan_hound: 156

INFO:root: Numero de imágenes de la clase n02088238-basset: 92

INFO:root: Numero de imágenes de la clase n02088364-beagle: 112

INFO:root: Numero de imágenes de la clase n02088466-bloodhound: 104

INFO:root: Numero de imágenes de la clase n02088632-bluetick: 88

INFO:root: Numero de imágenes de la clase n02089078-black-and-tan_coonhound: 76

INFO:root: Numero de imágenes de la clase n02089867-Walker_hound: 70

INFO:root: Numero de imágenes de la clase n02089973-English_foxhound: 74

INFO:root: Numero de imágenes de la clase n02090379-redbone: 65

INFO:root: Numero de imágenes de la clase n02090622-borzoi: 68

INFO:root: Numero de imágenes de la clase n02090721-Irish_wolfhound: 135

INFO:root: Numero de imágenes de la clase n02091032-Italian_greyhound: 99

INFO:root: Numero de imágenes de la clase n02091134-whippet: 104

INFO:root: Numero de imágenes de la clase n02091244-Ibizan_hound: 105

INFO:root: Numero de imágenes de la clase n02091467-Norwegian_elkhound: 113

INFO:root: Numero de imágenes de la clase n02091635-otterhound: 68

INFO:root: Numero de imágenes de la clase n02091831-Saluki: 117

INFO:root: Numero de imágenes de la clase n02092002-Scottish_deerhound: 149

INFO:root: Numero de imágenes de la clase n02092339-Weimaraner: 77

INFO:root: Numero de imágenes de la clase n02093256-Staffordshire_bullterrier: 72

INFO:root: Numero de imágenes de la clase n02093428-American_Staffordshire_terrier: 81

INFO:root: Numero de imágenes de la clase n02093647-Bedlington_terrier: 99

INFO:root: Numero de imágenes de la clase n02093754-Border_terrier: 89



INFO:root: Numero de imágenes de la clase n02093859-Kerry_blue_terrier: 96
INFO:root: Numero de imágenes de la clase n02093991-Irish_terrier: 86
INFO:root: Numero de imágenes de la clase n02094114-Norfolk_terrier: 89
INFO:root: Numero de imágenes de la clase n02094258-Norwich_terrier: 102
INFO:root: Numero de imágenes de la clase n02094433-Yorkshire_terrier: 81
INFO:root: Numero de imágenes de la clase n02095314-wire-haired_fox_terrier: 74
INFO:root: Numero de imágenes de la clase n02095570-Lakeland_terrier: 114
INFO:root: Numero de imágenes de la clase n02095889-Sealyham_terrier: 119
INFO:root: Numero de imágenes de la clase n02096051-Airedale: 119
INFO:root: Numero de imágenes de la clase n02096177-cairn: 114
INFO:root: Numero de imágenes de la clase n02096294-Australian_terrier: 113
INFO:root: Numero de imágenes de la clase n02096437-Dandie_Dinmont: 97
INFO:root: Numero de imágenes de la clase n02096585-Boston_bull: 99
INFO:root: Numero de imágenes de la clase n02097047-miniature_schnauzer: 71
INFO:root: Numero de imágenes de la clase n02097130-giant_schnauzer: 74
INFO:root: Numero de imágenes de la clase n02097209-standard_schnauzer: 72
INFO:root: Numero de imágenes de la clase n02097298-Scotch_terrier: 75
INFO:root: Numero de imágenes de la clase n02097474-Tibetan_terrier: 123
INFO:root: Numero de imágenes de la clase n02097658-silky_terrier: 100
INFO:root: Numero de imágenes de la clase n02098105-soft-coated_wheaten_terrier: 73
INFO:root: Numero de imágenes de la clase n02098286-West_Highland_white_terrier: 86
INFO:root: Numero de imágenes de la clase n02098413-Lhasa: 103
INFO:root: Numero de imágenes de la clase n02099267-flat-coated_retriever: 69
INFO:root: Numero de imágenes de la clase n02099429-curly-coated_retriever: 68
INFO:root: Numero de imágenes de la clase n02099601-golden_retriever: 67
INFO:root: Numero de imágenes de la clase n02099712-Labrador_retriever: 88
INFO:root: Numero de imágenes de la clase n02099849-Chesapeake_Bay_retriever: 84
INFO:root: Numero de imágenes de la clase n02100236-German_short-haired_pointer: 69
INFO:root: Numero de imágenes de la clase n02100583-vizsla: 71
INFO:root: Numero de imágenes de la clase n02100735-English_setter: 78
INFO:root: Numero de imágenes de la clase n02100877-Irish_setter: 72
INFO:root: Numero de imágenes de la clase n02101006-Gordon_setter: 70
INFO:root: Numero de imágenes de la clase n02101388-Brittany_spaniel: 69
INFO:root: Numero de imágenes de la clase n02101556-clumber: 67
INFO:root: Numero de imágenes de la clase n02102040-English_springer: 76
INFO:root: Numero de imágenes de la clase n02102177-Welsh_springer_spaniel: 67
INFO:root: Numero de imágenes de la clase n02102318-cocker_spaniel: 76
INFO:root: Numero de imágenes de la clase n02102480-Sussex_spaniel: 68
INFO:root: Numero de imágenes de la clase n02102973-Irish_water_spaniel: 67
INFO:root: Numero de imágenes de la clase n02104029-kuvasz: 67
INFO:root: Numero de imágenes de la clase n02104365-schipperke: 71
INFO:root: Numero de imágenes de la clase n02105056-groenendael: 67
INFO:root: Numero de imágenes de la clase n02105162-malinois: 67
INFO:root: Numero de imágenes de la clase n02105251-briard: 69
INFO:root: Numero de imágenes de la clase n02105412-kelpie: 70
INFO:root: Numero de imágenes de la clase n02105505-komondor: 71
INFO:root: Numero de imágenes de la clase n02105641-Old_English_ssheepdog: 86



INFO:root: Numero de imágenes de la clase n02105855-Shetland_sheepdog: 74
INFO:root: Numero de imágenes de la clase n02106030-collie: 70
INFO:root: Numero de imágenes de la clase n02106166-Border_collie: 67
INFO:root: Numero de imágenes de la clase n02106382-Bouvier_des_Flandres: 67
INFO:root: Numero de imágenes de la clase n02106550-Rottweiler: 69
INFO:root: Numero de imágenes de la clase n02106662-German_shepherd: 69
INFO:root: Numero de imágenes de la clase n02107142-Doberman: 67
INFO:root: Numero de imágenes de la clase n02107312-miniature_pinscher: 101
INFO:root: Numero de imágenes de la clase n02107574-Greater_Swiss_Mountain_dog: 85
INFO:root: Numero de imágenes de la clase n02107683-Bernese_mountain_dog: 135
INFO:root: Numero de imágenes de la clase n02107908-Appenzeller: 68
INFO:root: Numero de imágenes de la clase n02108000-EntleBucher: 119
INFO:root: Numero de imágenes de la clase n02108089-boxer: 68
INFO:root: Numero de imágenes de la clase n02108422-bull_mastiff: 73
INFO:root: Numero de imágenes de la clase n02108551-Tibetan_mastiff: 69
INFO:root: Numero de imágenes de la clase n02108915-French_bulldog: 76
INFO:root: Numero de imágenes de la clase n02109047-Great_Dane: 73
INFO:root: Numero de imágenes de la clase n02109525-Saint_Bernard: 87
INFO:root: Numero de imágenes de la clase n02109961-Eskimo_dog: 67
INFO:root: Numero de imágenes de la clase n02110063-malamute: 95
INFO:root: Numero de imágenes de la clase n02110185-Siberian_husky: 109
INFO:root: Numero de imágenes de la clase n02110627-affenpinscher: 67
INFO:root: Numero de imágenes de la clase n02110806-basenji: 126
INFO:root: Numero de imágenes de la clase n02110958-pug: 117
INFO:root: Numero de imágenes de la clase n02111129-Leonberg: 127
INFO:root: Numero de imágenes de la clase n02111277-Newfoundland: 112
INFO:root: Numero de imágenes de la clase n02111500-Great_Pyrenees: 130
INFO:root: Numero de imágenes de la clase n02111889-Samoyed: 135
INFO:root: Numero de imágenes de la clase n02112018-Pomeranian: 136
INFO:root: Numero de imágenes de la clase n02112137-chow: 113
INFO:root: Numero de imágenes de la clase n02112350-keeshond: 75
INFO:root: Numero de imágenes de la clase n02112706-Brabancon_griffon: 70
INFO:root: Numero de imágenes de la clase n02113023-Pembroke: 98
INFO:root: Numero de imágenes de la clase n02113186-Cardigan: 72
INFO:root: Numero de imágenes de la clase n02113624-toy_poodle: 68
INFO:root: Numero de imágenes de la clase n02113712-miniature_poodle: 72
INFO:root: Numero de imágenes de la clase n02113799-standard_poodle: 76
INFO:root: Numero de imágenes de la clase n02113978-Mexican_hairless: 72
INFO:root: Numero de imágenes de la clase n02115641-dingo: 73
INFO:root: Numero de imágenes de la clase n02115913-dhole: 67
INFO:root: Numero de imágenes de la clase n02116738-African_hunting_dog: 86
INFO:root: Numero de imágenes de todas las clases en total: 10620



CONFIGURACIÓN DEL SERVIDOR AMAZON LINUX (DEEP LEARNING | VERSIÓN 33.0)

Tras conocer en detalle la cantidad de datos que se iban a utilizar, y tras realizar varias pruebas en local, fue realmente necesario el uso de un servidor de Amazon Web Service (AWS), de pago, para el entrenamiento del modelo. Ya que, es importante tener en cuenta que, por la gran cantidad de datos seleccionados, no era viable entrenar los modelos en la máquina local. Por esta razón, se ha seleccionado una máquina de 8 cores, 32 GB de RAM y 100 GB de memoria, correspondiente con el servidor de Amazon Linux Versión 33.0.

De manera que, una vez levantado el servidor, se ha procedido a su configuración para la elaboración de los modelos pertinentes. Para ello, se ha hecho uso del cliente SSH PuTTY para generar la key *.ppk* que nos servirá junto a la *ip* para conectarnos con el terminal de MobaXterm, a la máquina en cuestión. Además de todo ello, es necesaria la descarga e instalación de Anaconda, que incluye la aplicación jupyter notebooks que utilizaremos para el diseño de los modelos.

El procedimiento realizado para la descarga y configuración del servidor es la siguiente:

Paso 1: Descargar Anaconda en la instancia EC2 de la máquina.

```
Wget https://repo.continuum.io/archive/Anaconda3-4.2.0-Linux-x86\_64.sh
```

Paso 2: Instalar Anaconda.

```
Bash Anaconda3-4.2.0-Linux-x86_64.sh
```

Paso 3: Creamos una password para el jupyter notebook.

```
Ipython
```

```
from Ipython.lib import passwd  
  
passwd()
```

```
Enter password: [Create password and press enter] Verify password: [Press enter]
```

Paso 4: Creamos config profile

```
mkdir certs
```

```
cd certs
```

```
openssl req -x509 -nodes -days 365 -newkey rsa:1024 -keyout mycert.pem -out mycert.pem
```

Paso 5: Configuramos Jupyter

```
cd ~/.jupyter/
```



```
vi jupyter_notebook_config.py
```

Añadimos lo siguiente en el .py:

```
c = get_config()  
# Kernel config  
c.IPKernelApp.pylab = 'inline' # if you want plotting support always in your notebook  
# Notebook config  
c.NotebookApp.certfile = u'/home/12ctiva/certs/mycert.pem' #location of your certificate file  
c.NotebookApp.ip = '0.0.0.0'  
c.NotebookApp.open_browser = False #so that the ipython notebook does not open a browser  
by default  
c.NotebookApp.password = u'sha1:98ff0e580111:12798c72623a6eecd54b51c006b1050f0ac1a62d'  
#the encrypted password we generated above  
# Set the port to 8888, the port we set up in the AWS EC2 set-up  
c.NotebookApp.port = 8888
```

Posteriormente, pulsamos:

```
esc  
shift-z
```

Paso 6: Creamos carpetas para los notebooks

```
cd ~
```

```
mkdir Notebooks
```

```
cd Notebooks
```

Paso 7: Creamos un nuevo screen

```
screen
```

Paso 8: Lanzamos Jupyter Notebooks

```
sudo chown $USER:$USER /home/12ctiva/certs/mycert.pem
```

```
jupyter notebook
```

Paso 9: Accedemos a jupyter por internet con el dns de la máquina de Amazon.

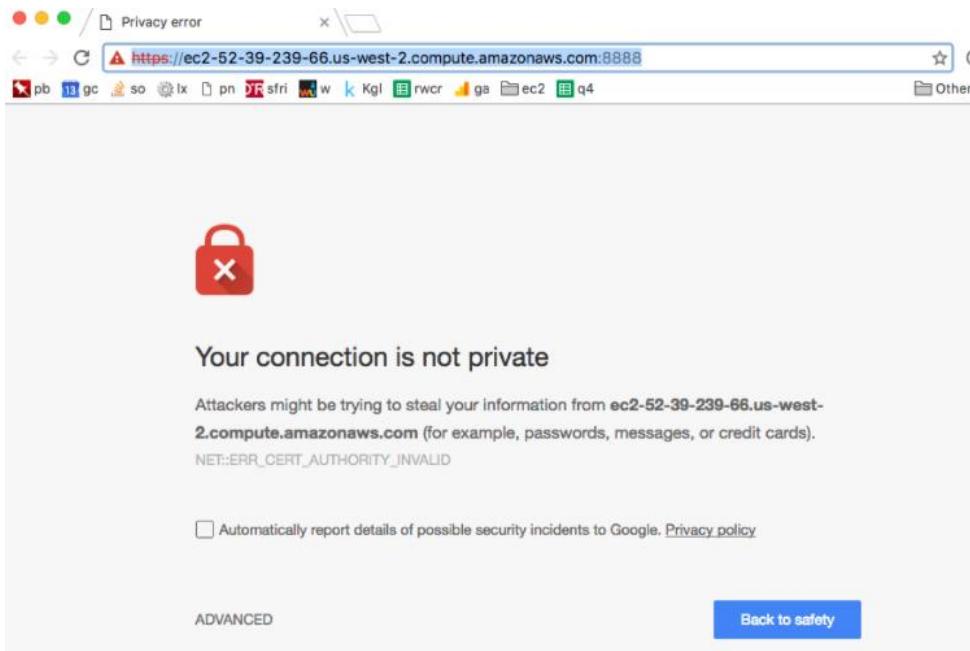


Figura 5. Acceso a jupyter por internet

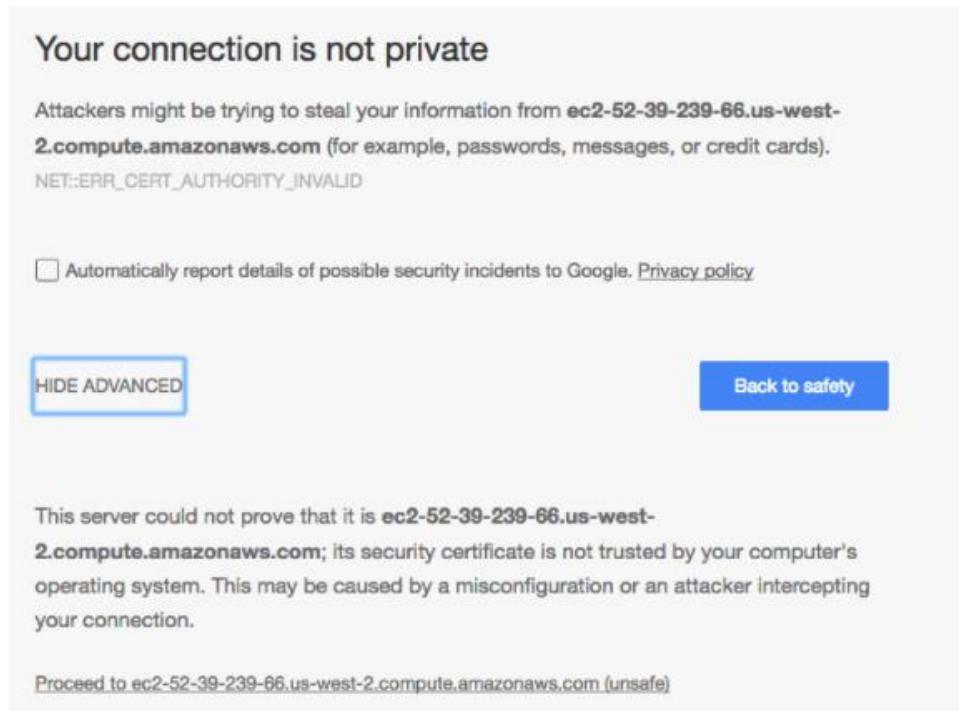


Figura 6. Acceso a jupyter por internet

A continuación, aparece una ventana en la que es necesario introducir la contraseña generada en el paso 3 del presente Anexo.



Password: Log in



Paso 10: Cargamos el dataset en la ruta: /home/ec2-user/images.

Durante el proceso de diseño, se han realizado varias pruebas en local, con menos cantidad de datos. De manera que, se ha generado un primer notebook como borrador, en local, que posteriormente se ha cargado en el servidor para realizar las pruebas con la información completa del Dataset utilizado, y, de esta forma, obtener un notebook definitivo.

Paso 11: Configuración e instalación de las librerías necesarias en el kernel del notebook en cuestión.

ESTUDIOS DE LOS MEJORES MODELOS CNNs

Para conseguir el objetivo propuesto, se van a elegir los mejores modelos preentrenados. Todo ello se va a realizar comparando el comportamiento de los modelos que nos ofrece keras. De manera que, va a ser necesario observar *Top-1 Accuracy* y *Top-5 Accuracy*, de los modelos proporcionados en la página oficial de keras (Modelos Keras, s.f.), para la elección de tres de ellos. Dichos modelos, también se pueden observar en la tabla 1, que se observa a continuación:

Model	Size	Top-1 Accuracy	Top-5 Accuracy	Parameters	Depth
Xception	88 MB	0.790	0.945	22.910.480	126
VGG16	582 MB	0.713	0.901	138.357.544	23
VGG19	549 MB	0.713	0.900	143.667.240	26
ResNet50	98 MB	0.749	0.921	25.636.712	-
ResNet101	171 MB	0.764	0.928	44.707.176	-
ResNet152	232 MB	0.766	0.931	60.419.944	-
ResNet50V2	98 MB	0.760	0.930	25.613.800	-
ResNet101V2	171 MB	0.772	0.938	44.675.560	-
ResNet152V2	232 MB	0.780	0.942	60.380.648	-
InceptionV3	92 MB	0.779	0.937	23.851.784	159
InceptionResNetV2	215 MB	0.803	0.953	55.873.763	572
MobileNet	16 MB	0.704	0.895	4.253.864	88
MobileNetV2	14 MB	0.713	0.901	3.538.984	88
DenseNet121	33 MB	0.750	0.923	8.062.504	121
DenseNet169	57 MB	0.762	0.932	14.307.880	169
DenseNet201	80 MB	0.773	0.936	20.242.984	201
NASNetMobile	23 MB	0.744	0.919	5.326.716	-
NASNetLarge	343 MB	0.825	0.960	88.949.818	-

Tabla 1. Modelos obtenidos de la página oficial de Keras.

Por lo que, teniendo en cuenta la información anterior, se van a usar los siguientes modelos preentrenados: VGG16, InceptionResNetV2, MobileNetV2. De manera que, se entrenarán únicamente las capas nuevas de cada modelo, y, posteriormente, se escogerá el modelo que presente mayor accuracy para realizar las predicciones con el conjunto de prueba.



DEFINICIÓN Y ENTRENAMIENTO DE LOS MODELOS

En el presente apartado se procederá a la definición y entrenamiento de los modelos seleccionados: *VGG16*, *InceptionResNetV2*, y *MobileNetV2*.

- Definición de los modelos.

De manera que, es necesario tener en cuenta que, al ser modelos preentrenados, será necesario eliminar la última capa de cada uno de ellos para adaptarla a los requerimientos de nuestro objetivo principal. Además, no será necesario entrenar las capas ya entrenadas, por lo que sólo se entrenaran las capas añadidas. De manera que, por cada modelo, se tienen las siguientes capas añadidas:

- **Capa añadida con el output de la última capa**, para la obtención promedio de las dimensiones espaciales. Para ello, se ha utilizado la función *GlobalAveragePooling2D*.
- **Capa con 512 nodos en la que se utilizará el rectificador lineal *activation relu***, que busca eliminar los valores negativos y dejar los positivos tal y como entran, es la función de activación más usada en deep learning y, especialmente, en los trabajos con imágenes.
- **Última capa densa, con 120 nodos**. Es la capa que identificará el tipo de raza de perro que tenemos en la imagen según los resultados que obtiene de las capas anteriores de la red. Esta capa utilizará la función *softmax*, que nos permite obtener la probabilidad de a qué clase de raza pertenece la imagen a identificar.

Los pasos que se han seguido para implementación de las capas comentadas se definen a continuación:

- **Paso 1:** Eliminamos la última capa del modelo preentrenado estableciendo el atributo *include_top* a False.
- **Paso 2:** Añadimos la primera capa utilizando el output del modelo pre-entrenado junto con la función *GlobalAveragePooling2D* para aplicar la agrupación promedio de las dimensiones espaciales.
- **Paso 3:** Añadimos la segunda capa con 512 nodos en ella y se aplicará la *activation relu* en la capa. Esta función es un rectificador lineal que busca eliminar los valores negativos y dejar los positivos tal y como entran, es la función de activación más usada en el 16cti learning y, especialmente, en los trabajos con imágenes.
- **Paso 4:** Añadimos la última capa densa, utilizando la función Dense, donde se usan los parámetros *units* y *activation*. Estos dos parámetros obtendrán el valor 120 y ‘softmax’. Estos valores son debidos al número de clases que tenemos que identificar en nuestro dataset y, por el otro lado, la función *softmax*, nos permite obtener una probabilidad de a qué clase de raza pertenece la imagen a identificar, es decir, es la capa que identificará el tipo de raza de perro que tenemos en la imagen según los resultados que obtiene de las capas anteriores de la red.

Ejemplos de la implementación descrita en los modelos utilizados:

➤ **Modelo VGG16**

```
#Modelo preentrenado VGG16
logging.info("Obtenemos el modelo base VGG16 sin la ultima capa")
modelVGG16_base=VGG16(weights='imagenet',include_top=False)
modelVGG16_base.trainable=False
x=modelVGG16_base.output
#Primera capa con el output de la ultima capa
logging.info("Primera capa con el output de la ultima capa")
x=GlobalAveragePooling2D()(x)
#dense Layer 3 crea una nueva capa oculta con 512 nodos con activacion relu
logging.info("Segunda capa oculta con 512 nodos con activacion relu")
x=Dense(512,activation='relu')(x)
#Creamos la ultima capa con 120 neuronas
#Las 120 razas de perros que queremos identificar
#y aplicamos la activacion softmax
logging.info("Ultima capa")
preds=Dense(120,activation='softmax')(x) #final layer with softmax activation
#Creamos el modelo con las nuevas capas
logging.info("Generamos el modelo con las capas nuevas")
modelVGG16=Model(inputs=modelVGG16_base.input,outputs=preds)
```

Figura 7. Modelo preentrenado VGG16

➤ **Modelo InceptionResNetV2**

```
#Modelo preentrenado InceptionResNetV2
logging.info("Obtenemos el modelo base InceptionResNetV2 sin la ultima capa")
modelInceptionResNetV2_base = keras.applications.resnet_v2.ResNet50V2(weights='imagenet', include_top=False)
modelInceptionResNetV2_base.trainable=False
x=modelInceptionResNetV2_base.output
logging.info("Primera capa con el output de la ultima capa")
x=GlobalAveragePooling2D()(x)
#dense Layer 3 crea una nueva capa oculta con 512 nodos con activacion relu
logging.info("Segunda capa oculta con 512 nodos con activacion relu")
x=Dense(512,activation='relu')(x)
#Crea la ultima capa con 3 nodos y activacion softmax
logging.info("Ultima capa")
preds=Dense(120,activation='softmax')(x) #final layer with softmax activation
logging.info("Generamos el modelo con las capas nuevas")
modelInceptionResNetV2=Model(inputs=modelInceptionResNetV2_base.input,outputs=preds)
```

Figura 8. Modelo InceptionResNetV2

➤ **Modelo MobileNetV2**

```
#Modelo preentrenado MobileNetV2
logging.info("Obtenemos el modelo base MobileNetV2 sin la ultima capa")
modelMobileNetV2_base = keras.applications.mobilenet_v2.MobileNetV2(weights='imagenet', include_top=False)
modelMobileNetV2_base.trainable=False
x=modelMobileNetV2_base.output
logging.info("Primera capa con el output de la ultima capa")
x=GlobalAveragePooling2D()(x)
logging.info("Segunda capa oculta con 512 nodos con activacion relu")
#dense Layer 3 crea una nueva capa oculta con 512 nodos con activacion relu
x=Dense(512,activation='relu')(x) |
#Crea la ultima capa con 3 nodos y activacion softmax
logging.info("Ultima capa")
preds=Dense(120,activation='softmax')(x) #final layer with softmax activation
logging.info("Generamos el modelo con las capas nuevas")
modelMobileNetV2=Model(inputs=modelMobileNetV2_base.input,outputs=preds)
```

Figura 9. Modelo preentrenado MobileNetV2

Por otro lado, tal y como se ha comentado previamente, se van a entrenar únicamente las capas añadidas, para ello, se ha utilizado el *transfer Learning*, que consiste en tomar las

características aprendidas en un problema y aprovecharlas para un problema nuevo similar. De manera que, se reutilizan las características antiguas en predicciones para un nuevo conjunto de datos. Esto es posible gracias al atributo booleano que tienen todas capas & modelos: `layer.trainable`. De manera que, únicamente se tienen que poner a false las capas que no se quieran entrenar, y a true las nuevas entrenables.

➤ **Modelo ModelVGG16**

```
logging.info("Establecemos las capas del modelo preentrenado a NO entrenables")
for layer in modelVGG16.layers[:19]:
    layer.trainable=False

logging.info("Establecemos las capas nuevas a entrenables")
for layer in modelVGG16.layers[19:]:
    layer.trainable=True

INFO:root:Establecemos las capas del modelo preentrenado a NO entrenables
INFO:root:Establecemos las capas nuevas a entrenables
```

Figura 10. Establecemos las capas nuevas a entrenables y a no entrenables para el Modelo ModelVGG16.

➤ **Modelo ModelMobileNetV2**

```
logging.info("Establecemos las capas del modelo preentrenado a NO entrenables")
for layer in modelMobileNetV2.layers[:155]:
    layer.trainable=False
logging.info("Establecemos las capas nuevas a entrenables")
for layer in modelMobileNetV2.layers[155:]:
    layer.trainable=True

INFO:root:Establecemos las capas del modelo preentrenado a NO entrenables
INFO:root:Establecemos las capas nuevas a entrenables
```

Figura 11. Establecemos las capas nuevas a entrenables y a no entrenables para el Modelo ModelMobileNetV2.

➤ **Modelo ModelInceptionResNetV2**

```
logging.info("Establecemos las capas del modelo preentrenado a NO entrenables")
for layer in modelInceptionResNetV2.layers[:190]:
    layer.trainable=False
logging.info("Establecemos las capas nuevas a entrenables")
for layer in modelInceptionResNetV2.layers[190:]:
    layer.trainable=True

INFO:root:Establecemos las capas del modelo preentrenado a NO entrenables
INFO:root:Establecemos las capas nuevas a entrenables
```

Figura 12. Establecemos las capas nuevas a entrenables y a no entrenables para el Modelo ModelInceptionResNetV2.

● Preentrenamiento y entrenamiento de las capas añadidas

Los datos de las imágenes del conjunto de entrenamiento, no se pueden leer ni convertir directamente en tensores. Sin embargo, Keras proporciona métodos incorporados que pueden realizar esta tarea fácilmente. Por tanto, para el preentrenamiento de los modelos se ha utilizado la clase `ImageDataGenerator` y el `train_generator`, que nos permiten trabajar con la información de las



imágenes, y, además, van a ser muy útiles para cambiar el tamaño de las imágenes, voltearlas, etc. De esta manera, agregamos más entrenamiento a los datos, para evitar un ajuste excesivo.

A continuación, se muestra la tabla 2 y la tabla 3, con las características propias de las clases utilizadas en el preentrenamiento de los modelos.

ImageDataGenerator	
Genere lotes de datos de imágenes de tensores con aumento de datos en tiempo real. Los datos se repetirán (en lotes).	
preprocessing_function	Función que se aplicará en cada entrada. La función se ejecutará después de que la imagen cambie de tamaño y aumente. La función debe tomar un argumento: una imagen (tensor Numpy con rango 3) y debe generar un tensor Numpy con la misma forma
zoom_range	Flotante o [inferior, superior]. Rango para zoom aleatorio. Si es un flotante, [inferior, superior] = [1-zoom_range, 1 + zoom_range]
horizontal_flip	Booleano. Voltee las entradas de forma aleatoria horizontalmente.
rango_rotación	Int. Rango de grados para rotaciones aleatorias

Tabla 2. Clase ImageDataGenerator

Train Generator	
Path	El directorio debe establecerse en la ruta donde están presentes sus "n" clases de carpetas.
Target_size	Es el tamaño de sus imágenes de entrada, cada imagen cambiará de tamaño a este tamaño
color_mode	Si la imagen es en blanco y negro o en escala de grises, configure "escala de grises" o si la imagen tiene tres canales de color, configure "rgb"
batch_size	No. de imágenes que se generarán desde el generador por lote
class_mode	Establezca "binario" si solo tiene dos clases para predecir, si no está establecido en "categórico", en caso de que esté desarrollando un sistema Autoencoder, tanto la entrada como la salida probablemente sean la misma imagen, para este caso, establezca a "entrada".
shuffle	Establezca True si desea mezclar el orden de la imagen que se está generando, de lo contrario, establezca False

Tabla 3. Train generator

Ejemplo código para el modelo VGG16:

```
logging.info("Utilizando ImageDataGenerator")
train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_vgg16,
    zoom_range=0.2,
    rotation_range = 5,
    horizontal_flip=True)

logging.info("Generando el train_generator")
train_generator=train_datagen.flow_from_directory(path, |
                                                    target_size=(256,256),
                                                    # default parameters
                                                    color_mode='rgb',
                                                    batch_size=4,
                                                    class_mode='categorical',
                                                    shuffle=True)
```

Figura 13. Preentrenamiento del modelo VGG16.

Posteriormente, se compilan los modelos y se hace uso de la función `fit_generator` para proceder al entrenamiento de los mismos. A continuación, se pueden observar las características de las funciones utilizadas para la compilación y entrenamiento del modelo, en la tabla 4, y tabla 5. Además, también se observa un ejemplo de código utilizado para el entrenamiento del modelo VGG16 en la figura 13.

Compile Method	
Optimizador	Cadena (nombre del optimizador) o instancia del optimizador. Consulte <code>tf.keras.optimizers</code> .
Loss	String (nombre de la función objetivo), función objetivo o instancia <code>tf.keras.losses.Loss</code> . Ver <code>tf.keras.losses</code> .
Metrics	Lista de métricas que el modelo evaluará durante el entrenamiento y las pruebas. Cada uno de estos puede ser una cadena (nombre de una función incorporada), función o una instancia de <code>tf.keras.metrics.Metric</code> . Consulte <code>tf.keras.metrics</code> .

Tabla 4. Compile Method

fit_generator	
Generator	Un generador cuya salida debe ser una lista de la forma: - (insumos, objetivos) - (entrada, objetivos, pesos_muestra) una sola salida del generador hace un solo lote y, por lo tanto, todas las matrices en la lista deben tener una longitud igual al tamaño del lote
steps_per_epoch	Especifica el número total de pasos tomados del generador tan pronto como termina una época y comienza la siguiente. Podemos calcular el valor de <code>steps_per_epoch</code> como el número total de muestras en su conjunto de datos dividido por el tamaño del lote
Épocas	Un número entero y número de épocas para las que queremos entrenar nuestro modelo

Tabla 5. Fit_generator

A continuación, se muestra el código implementado para el entrenamiento del modelo VGG16.

```
logging.info("Compilando el modelo")
modelVGG16.compile(optimizer='Adam', loss='categorical_crossentropy', metrics=['accuracy'])
logging.info("Comienzo de entrenamiento")
step_size_train=train_generator.n//train_generator.batch_size
model = modelVGG16.fit_generator(generator=train_generator,
                                    steps_per_epoch=step_size_train,
                                    epochs=20)
```

Figura 14. Compilamos y entrenamos el modelo VGG16

COMPARACIÓN DE LOS MODELOS Y SELECCIÓN DEL MEJOR

A continuación, se muestra una tabla con la precisión obtenida por los diferentes modelos entrenados, únicamente se muestra la precisión obtenida en la etapa 20, es decir, la última precisión obtenida:

Modelo	Precisión
VGG16	0.87
InceptionResNetV2	0.87
MobileNetV2	0.92

Figura 15. Comparación de los modelos seleccionados.

Se observa como claramente el modelo que utiliza la red preentrenada **MobileNetV2** obtiene una precisión superior a todos los demás modelos estudiados, **por lo que elegimos este modelo como ganador**. A continuación, se puede observar el resultado de la precisión obtenida en cada etapa por cada modelo:

- **Modelo VGG16**

Resultados del entrenamiento:

Epoch 1/20

2655/2655 [=====] - 1059s 399ms/step - loss: 0.6733 - accuracy: 0.82
83

Epoch 2/20

2655/2655 [=====] - 1059s 399ms/step - loss: 0.7151 - accuracy: 0.82
83

Epoch 3/20

2655/2655 [=====] - 1059s 399ms/step - loss: 0.6755 - accuracy: 0.82
87

Epoch 4/20

2655/2655 [=====] - 1062s 400ms/step - loss: 0.6658 - accuracy: 0.83
35

Epoch 5/20

2655/2655 [=====] - 1062s 400ms/step - loss: 0.6374 - accuracy: 0.84
04

Epoch 6/20

2655/2655 [=====] - 1064s 401ms/step - loss: 0.6546 - accuracy: 0.84
23

Epoch 7/20

2655/2655 [=====] - 1065s 401ms/step - loss: 0.6296 - accuracy: 0.84
89

Epoch 8/20

2655/2655 [=====] - 1065s 401ms/step - loss: 0.6355 - accuracy: 0.84
75

Epoch 9/20

2655/2655 [=====] - 1063s 400ms/step - loss: 0.6146 - accuracy: 0.85
31

Epoch 10/20



2655/2655 [=====] - 1064s 401ms/step - loss: 0.6247 - accuracy: 0.85
68
Epoch 11/20
2655/2655 [=====] - 1066s 401ms/step - loss: 0.6350 - accuracy: 0.85
39
Epoch 12/20
2655/2655 [=====] - 1065s 401ms/step - loss: 0.6159 - accuracy: 0.85
74
Epoch 13/20
2655/2655 [=====] - 1063s 401ms/step - loss: 0.6508 - accuracy: 0.85
72
Epoch 14/20
2655/2655 [=====] - 1065s 401ms/step - loss: 0.5847 - accuracy: 0.86
34
Epoch 15/20
2655/2655 [=====] - 1063s 400ms/step - loss: 0.5974 - accuracy: 0.86
41
Epoch 16/20
2655/2655 [=====] - 1063s 400ms/step - loss: 0.6036 - accuracy: 0.86
69
Epoch 17/20
2655/2655 [=====] - 1062s 400ms/step - loss: 0.6079 - accuracy: 0.87
05
Epoch 18/20
2655/2655 [=====] - 1063s 400ms/step - loss: 0.6315 - accuracy: 0.86
34
Epoch 19/20
2655/2655 [=====] - 1062s 400ms/step - loss: 0.6147 - accuracy: 0.86
98
Epoch 20/20
2655/2655 [=====] - 1065s 401ms/step - loss: 0.5761 - accuracy: 0.87
66

- **Modelo InceptionResNetV2**

Resultados del entrenamiento:

Epoch 1/20
5310/5310 [=====] - 369s 69ms/step - loss: 0.9341 - accuracy: 0.7515
Epoch 2/20
5310/5310 [=====] - 369s 69ms/step - loss: 0.8763 - accuracy: 0.7664
Epoch 3/20
5310/5310 [=====] - 368s 69ms/step - loss: 0.8536 - accuracy: 0.7745
Epoch 4/20
5310/5310 [=====] - 367s 69ms/step - loss: 0.8376 - accuracy: 0.7866
Epoch 5/20
5310/5310 [=====] - 368s 69ms/step - loss: 0.7926 - accuracy: 0.7893
Epoch 6/20
5310/5310 [=====] - 367s 69ms/step - loss: 0.7865 - accuracy: 0.8005
Epoch 7/20
5310/5310 [=====] - 368s 69ms/step - loss: 0.7600 - accuracy: 0.8060
Epoch 8/20
5310/5310 [=====] - 370s 70ms/step - loss: 0.7289 - accuracy: 0.8182



Epoch 9/20

5310/5310 [=====] - 370s 70ms/step - loss: 0.7355 - accuracy: 0.8198

Epoch 10/20

5310/5310 [=====] - 368s 69ms/step - loss: 0.6922 - accuracy: 0.8336

Epoch 11/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.7020 - accuracy: 0.8298

Epoch 12/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.6821 - accuracy: 0.8394

Epoch 13/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.6946 - accuracy: 0.8432

Epoch 14/20

5310/5310 [=====] - 369s 70ms/step - loss: 0.6715 - accuracy: 0.8464

Epoch 15/20

5310/5310 [=====] - 369s 69ms/step - loss: 0.6331 - accuracy: 0.8549

Epoch 16/20

5310/5310 [=====] - 369s 69ms/step - loss: 0.6546 - accuracy: 0.8579

Epoch 17/20

5310/5310 [=====] - 369s 69ms/step - loss: 0.6190 - accuracy: 0.8623

Epoch 18/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.6068 - accuracy: 0.8651

Epoch 19/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.6195 - accuracy: 0.8654

Epoch 20/20

5310/5310 [=====] - 367s 69ms/step - loss: 0.6009 - accuracy: 0.8761

- **Modelo MobileNetV2**

Resultados del entrenamiento:

Epoch 1/20

5310/5310 [=====] - 213s 40ms/step - loss: 1.2467 - accuracy: 0.6503

Epoch 2/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.9041 - accuracy: 0.7337

Epoch 3/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.7751 - accuracy: 0.7717

Epoch 4/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.6829 - accuracy: 0.7987

Epoch 5/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.6420 - accuracy: 0.8146

Epoch 6/20

5310/5310 [=====] - 214s 40ms/step - loss: 0.5777 - accuracy: 0.8347

Epoch 7/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.5474 - accuracy: 0.8483

Epoch 8/20

5310/5310 [=====] - 214s 40ms/step - loss: 0.5126 - accuracy: 0.8565

Epoch 9/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.4898 - accuracy: 0.8695

Epoch 10/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.4614 - accuracy: 0.8771



Epoch 11/20

5310/5310 [=====] - 212s 40ms/step - loss: 0.4538 - accuracy: 0.8837

Epoch 12/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.4312 - accuracy: 0.8914

Epoch 13/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.4179 - accuracy: 0.8957

Epoch 14/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.4047 - accuracy: 0.9032

Epoch 15/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.4105 - accuracy: 0.9056

Epoch 16/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.3910 - accuracy: 0.9088

Epoch 17/20

5310/5310 [=====] - 214s 40ms/step - loss: 0.3958 - accuracy: 0.9169

Epoch 18/20

5310/5310 [=====] - 214s 40ms/step - loss: 0.3963 - accuracy: 0.9153

Epoch 19/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.3939 - accuracy: 0.9194

Epoch 20/20

5310/5310 [=====] - 213s 40ms/step - loss: 0.3874 - accuracy: 0.9234

Generación de un Apache Tomcat en el servidor de Amazon Linux de Deep Learning.

Ha sido necesario el despliegue de un tomcat en el servidor de amazon Linux para conectar el Servicio REST con la parte de Big Data. De manera que, cuando un usuario introduzca una imagen en la página web, el servicio REST, será el encargado de devolver una ruta concreta de la imagen. De manera que, el servicio Rest tendrá que preguntar al tomcat de amazon linux de Big Data, por la URL en cuestión. Dicha URL será recibida por el modelo generado, y, en consecuencia, se generará una respuesta con el tipo de raza de perro correspondiente a dicha imagen. De manera que, la respuesta será enviada desde el tomcat de big data, hasta el servicio rest, y éste hará posible que el usuario visualice el tipo de raza de perro por la consola del sitio web implementado. Todo ello, se puede visualizar en la imagen:

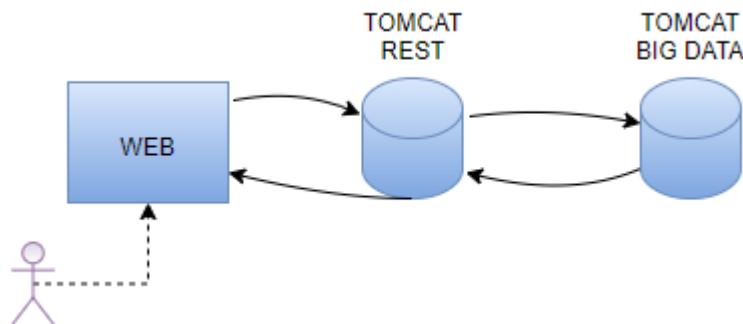


Figura 16. . Esquema sobre la estructura y conexión del servicio Rest y la aplicación Big Data

De manera que, cuando el usuario acceda a la parte web para identificar la raza de su perro:

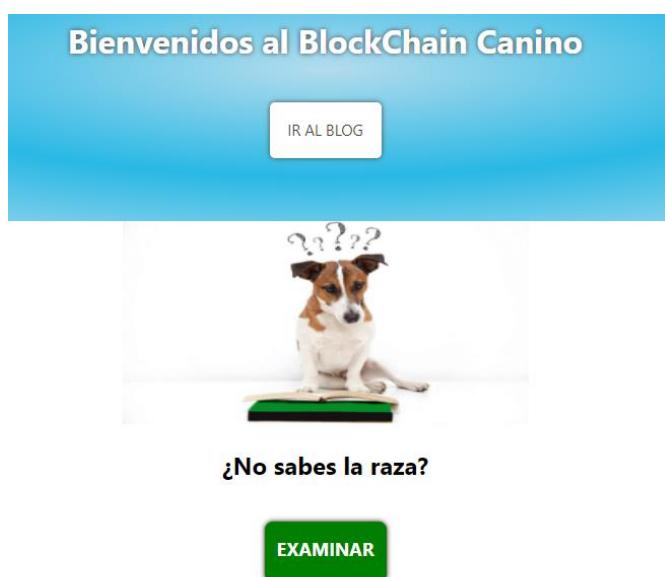


Figura 17. Sección de sitio web



Selecciona la imagen

n02085620_1346.jpg

¡Procede a identificar la raza!

Figura 18. Pasos necesarios en el sitio web desarrollado para la identificación de la raza de perro.



Figura 19. Obtención de la raza de perro

Predicciones con los datos de prueba

Después de haber seleccionado el mejor modelo, se procede a continuación a su evaluación, y a la elaboración de las predicciones del conjunto de prueba.

Para visualizar la precisión del modelo, se ha generado una gráfica que representa la precisión obtenida por cada época. De manera que, se comprueba que el modelo ganador **MobileNetV2** está alcanzado el 0.92 cuando se alcanza la etapa 20.

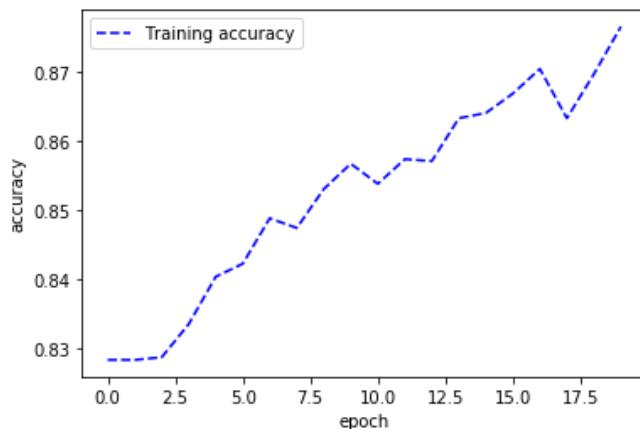


Figura 20. Training accuracy del modelo ganador MobileNetV2

Vamos a utilizar nuestro modelo para predecir las imágenes que hemos guardado anteriormente, las cuales no han sido utilizadas en el entrenamiento del modelo. Para ello, se ha generado una función *predict_image* que recibe como parámetro el path referente a la imagen de prueba. De manera que, esta función será la que reciba el path de la imagen que el usuario introduzca por el sitio web. A continuación, se observa el código realizado para la realización de esta función en la figura 17.

- Código implementado para la función *predict_image*:

```
class_dict = {v:k for k, v in train_generator.class_indices.items()}

def predict_image(path):
    img = image.load_img(path)
    img = img.resize((224, 224))
    data = expand_dims(image.img_to_array(img), 0)
    data = preprocess_mobilenet(data)
    preds = modelMobileNetV2.predict(data)
    pred = np.argmax(preds)
    pred = class_dict[pred]
    print(pred)
    return img|
```

Figura 21. Función *predict_image*

No obstante, la función comentada forma parte de un script que es capaz de procesar la imagen que recibe desde el rest. Dicho Script, se ejecuta dentro del tomcat del servicio de Big Data, ya que debe de cargar el modelo implementado, y dicho TomCat está configurado para la realización de las funciones o tareas específicas de Deep Learning. Es importante comentar que, debido principalmente a las configuraciones intrínsecas del servidor de big data para la ejecución del modelo, no ha sido viable el uso de un solo tomcat que se encargue del envío y recepción de



las peticiones por parte de blockchain y big data. Ya que, el tomcat del rest no presentaba la misma configuración.

A continuación, se puede observar el código referente al script implementado:

```
import sys
import os
from keras.models import load_model
import urllib.request
from PIL import Image
from keras.models import load_model
import urllib.request
from PIL import Image
import pandas as pd
import numpy as np
from numpy import expand_dims
import os
import sys
import re
import shutil
import keras
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.applications import MobileNet, VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.mobilenet import preprocess_input as preprocess_mobilenet
from tensorflow.keras.applications.densenet import preprocess_input as preprocess_densenet
from tensorflow.keras.applications.nasnet import preprocess_input as preprocess_nasnet
from tensorflow.keras.applications.vgg16 import preprocess_input as preprocess_vgg16
from tensorflow.keras.applications.inception_resnet_v2 import preprocess_input as preprocess_inceptionResNetV2
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import matplotlib.pyplot as plt
import logging
from random import randrange
import pydot
from keras.utils.vis_utils import plot_model
from IPython.display import Image
from IPython.core.display import HTML
import collections
import math
import os
import time
import tensorflow as fl
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications import VGG16
from tensorflow.keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Dropout,
GlobalAveragePooling2D
from tensorflow.keras.models import Model, Sequential
from PIL import Image
```



```
model = keras.models.load_model('/home/ec2-user/Notebooks/modelMobileNetV2.h5')
path = sys.argv[1]
print("El path es: ", path)
images = Image.open(urllib.request.urlopen(path))
images

def predict_image(path):
    img = path.resize((224, 224))
    data = expand_dims(path, 0)
    data = preprocess_mobilenet(data)
    preds = model.predict(data)
    pred = np.argmax(preds)
    return pred

indice = predict_image(images)

#Obtenemos las clases
classes = []
count = 0
data = []

datasetPath = "/home/ec2-user/images/Images"
logging.info(" Buscando la informacion del dataset")
for root, dirnames, filenames in os.walk(datasetPath):
    if(not classes):
        classes = dirnames
        continue
    data.append((classes[count],filenames))
    count+=1

print("raza:"+ classes[indice])
```

En resumen, estamos obteniendo buenos resultados por parte del algoritmo realizado para la generación de las predicciones. De manera que, se asegura con una probabilidad superior al 90%, de que el usuario no va a recibir como respuesta una raza de perro que no se corresponda con la raza del perro de la imagen introducida por el sitio web.



Conclusiones y líneas futuras

CONCLUSIONES

En el presente Anexo V, sobre Big Data Canino, se ha desarrollado un modelo de red neuronal convolucional con una precisión de 0.92. Para ello, se han llevado a cabo los siguientes pasos, coincidentes con el cumplimiento de los objetivos iniciales:

- Se ha utilizado un dataset de 22.580 imágenes, llamado Standford Dog Dataset. Obtenido del siguiente sitio web <https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>.
- Obtención del conjunto de datos de prueba y entrenamiento. Para ello, se ha desarrollado un script que introduce 83 imágenes de cada tipo de raza en una carpeta llamada Images_Test. Por lo tanto, finalmente se tienen 9960 imágenes destinadas al conjunto de prueba, y 10620, destinadas al conjunto de entrenamiento.
- Búsqueda de los mejores modelos preentrenados, y entrenamiento con distintos modelos. Se han evaluado los modelos presentes en la página oficial de Keras, y se han entrenado en un servidor de Amazon de pago para poder compararlos y seleccionar el mejor de ellos. Los modelos seleccionados han sido: VGG16, InceptionResNetV2 y MobileNetV2.
- Comparación de los modelos entrenados y obtención del mejor modelo. Se han comparado los tres modelos seleccionados y se ha seleccionado el modelo mobilenet, como mejor modelo, por tener mayor precisión.
- Realización de las predicciones con los datos de prueba, y análisis de los resultados.
- Generar un script que permita la obtención de la predicción de la imagen que el usuario introduzca en la página web.

LÍNEAS FUTURAS

- Generación e implementación de un proceso automático que corra en el servidor del Servicio Rest, cuyo principal objetivo sea entrenar el modelo a una hora concreta, de un día particular. De manera que, el modelo sería capaz de ir mejorándose o aprendiendo por sí sólo, con las nuevas imágenes introducidas por el usuario, a través de la página web.



Bibliografía

- ARTEAGA, G. J. (2015). *APLICACIÓN DEL APRENDIZAJE PROFUNDO (“DEEP LEARNING”) AL. SANTIAGO DE CALI .*
- Bagnato, J. I. (Noviembre de 2018). *Aprende Machine Learning*. Obtenido de
<https://www.aprendemachinelearning.com/como-funcionan-las-convolutional-neural-networks-vision-por-ordenador/>
- Cárdenas, I. R. (22 de Enero de 2018). *EEP LEARNING PARA LA DETECCIÓN DE PEATONES Y VEHÍCULOS*. Obtenido de
<http://148.215.1.182/bitstream/handle/20.500.11799/70995/tesisfinalRVC-ilovepdf-compressed%20%281%29.pdf?sequence=1&isAllowed=y>
- Egea, J. (1994). Redes neuronales: concepto, fundamento y aplicaciones en el laboratorio clínico. *Química clínica*, 13 (5): 221-228.
- García, E. M. (Septiembre de 2019). *TFG*. Obtenido de https://e-archivo.uc3m.es/bitstream/handle/10016/30357/TFG_Elena_Martinez_Garcia_2019.pdf?sequence=1&isAllowed=y
- J. A. Pérez-Carrasco, C. S.-G. (s.f.). *RED NEURONAL CONVOLUCIONAL*. Obtenido de
<https://idus.us.es/bitstream/handle/11441/79308/RED%20NEURONAL.pdf?sequence=1&isAllowed=y>
- Li, J. (s.f.). *kaggle*. Obtenido de <https://www.kaggle.com/jessicali9530/stanford-dogs-dataset>
- Modelos Keras*. (s.f.). Obtenido de <https://keras.io/api/applications/>,
- Pacheco, M. A. (Agosto de 2017). *Identificación de sistemas no lineales con redes neuronales convolucionales*. Obtenido de <https://www.ctrl.cinvestav.mx/~yuw/pdf/MaTesMLP.pdf>