

Guía de instalación

Puesta en producción

Introducción	3
Requerimientos mínimos	4
Software	4
Tecnologías utilizadas	4
Hardware	5
Arquitectura del sistema	5
Arquitectura de la solución	5
Modelado	7
Modelo de la base de datos	7
Esquema usuarios	8
Esquema proveedores	9
Esquema catalogos	10
Esquema proveedorRegistros	10
Esquema bitacora	11
Configuración del ambiente	11
Instalación de Docker	11
Instalación de Docker compose	14
Instalación de git	15
Instalación de MongoDB	16
Configuración de Docker Compose para MongoDB	16
Archivo de configuración de la base de datos (mongod.conf)	18
Archivo de configuración .env	21
Despliegue de la base de datos en Docker Compose	22
Pruebas de operación	23
Creación de usuarios	23
Creación de usuarios	23

Creación de usuario administrador	25
Creación y carga de los catálogos	27
Instalación del servidor OAuth2	30
Esquema de la base de datos	33
Usuarios	33
Clientes	33
Tokens	34
Pruebas de operación	36
Generar un token usando base 64	36
Generar un token a través del body	36
Respuesta	37
Instalación del Backend	37
Configuración de Docker Compose para backend	39
Pruebas de operación	40
Instalación del frontend	42
Configuración de Docker Compose para frontend	43
Pruebas de operación	45
Seguridad	46
Firewall	46
Certificado SSL	47
Contar con un dominio	48
Vincular la IP del servidor al dominio	50
Vincular el dominio al servidor	52
Implementar Let's Encrypt	55
Anexo 1 - Base de datos	61
Consola	61
Conexión	61
Operaciones - Base de Datos	63
Operaciones - Colecciones	64
Operaciones - Documentos	65
GUI Robo3T	66
Conexión	68
Operaciones	70

1. Introducción

El presente documento contiene la **Guía de instalación** para la puesta en producción del Sistema de Carga de Datos para el Sistema de Servidores Públicos que Intervienen en Procedimientos de Contratación (S2) y el Sistema de los Servidores Públicos y Particulares Sancionados (S3), dos de los seis sistemas de los cuales se conforma la Plataforma Digital Nacional (PDN).

El sistema de carga de datos permite a los generadores de datos transferir a las Secretarías Ejecutivas Anticorrupción Estatales los datos de los sistemas 2 y 3, que serán consultados desde la PDN a través de los mecanismos de comunicación desarrollados.

Las Secretarías Ejecutivas Anticorrupción Estatales fungirán como entidades concentradoras de los datos, sin embargo, el control y administración de los mismos será responsabilidad de los generadores/proveedores de datos.

El sistema de carga de información permite a los generadores de datos transferir a las Secretarías Ejecutivas Anticorrupción Estatales la información de los sistemas 2 y 3, que a través de los mecanismos de comunicación desarrollados se conectarán y se visualizarán los datos de estos sistemas en la PDN.

2. Requerimientos mínimos

A continuación se presentan los requerimientos mínimos de software y hardware para la puesta en marcha del Sistema de Carga de Datos del S2 y S3 de la PDN.

2.1. Software

Cualquier sistema operativo de Linux es soportado. Esta guía está basada en CentOS 7.x, en cualquier otra distribución de Linux, los comandos y repositorios pueden variar.

Sistema Operativo: CentOS Linux 7.x (Core) (**Recomendado**)

2.1.1. Tecnologías utilizadas

Tecnología	Versión	Descripción
Node.js	12.18.2	Entorno base de JavaScript, se usa como motor de ejecución para otras tecnologías del proyecto.
Express.js	4.16.3	Se usa como servidor para las solicitudes de API REST
React	17.0.1	Biblioteca JavaScript de código abierto diseñada para crear interfaces de usuario, con el objetivo de facilitar el desarrollo de aplicaciones en una sola página.
Mongoose	5.9.26	Funciona como biblioteca para realizar la conexión e interacción con la base de datos.
Webpack	4.17.2	Empaquetador de módulos.

Babel	7.0.0	Herramienta que nos permite transformar el código JS
Redux	4.0.0	Manejo del estado de las variables en React

2.2. Hardware

Las características de software y hardware especificadas son las recomendadas, pero bien el sistema puede trabajar con características inferiores, las únicas restricciones son que el sistema operativo sea CentOS 7 o Ubuntu, 1 CPU, 4 GB de RAM y 50 GB de espacio libre de disco, lo anterior puede resultar en un menor rendimiento.

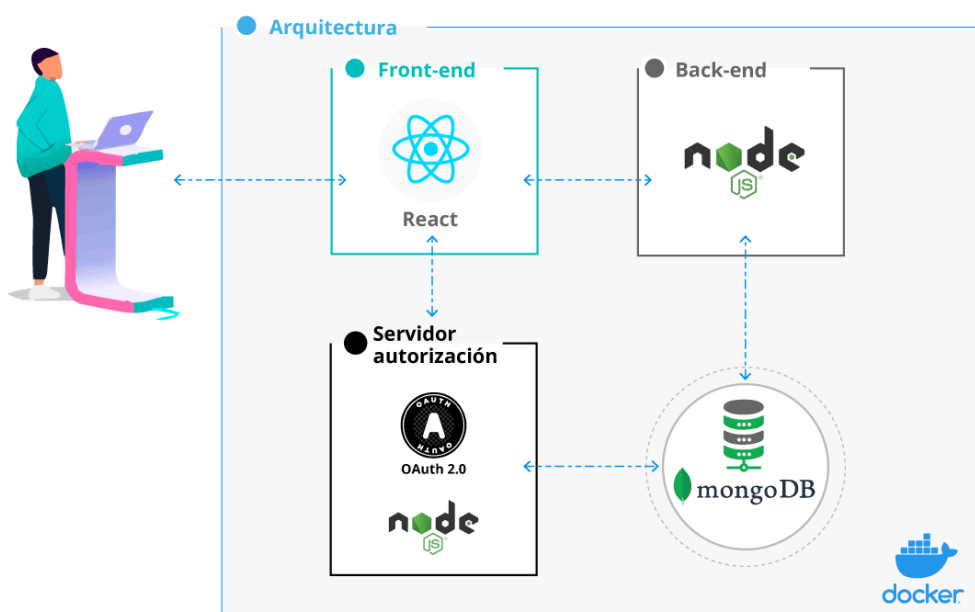
- **Procesador:** 4 CPU
- **Sistema Operativo:** CentOS 7.x
- **Memoria:** 8 GB RAM
- **Almacenamiento:** 50 GB Libres (aplicaciones/código)
- **Almacenamiento Base de datos:** 100 GB (inicial e incremental)
- **Transferencia:** 1 TB mensualmente (ambiente en nube)

3. Arquitectura del sistema

3.1. Arquitectura de la solución

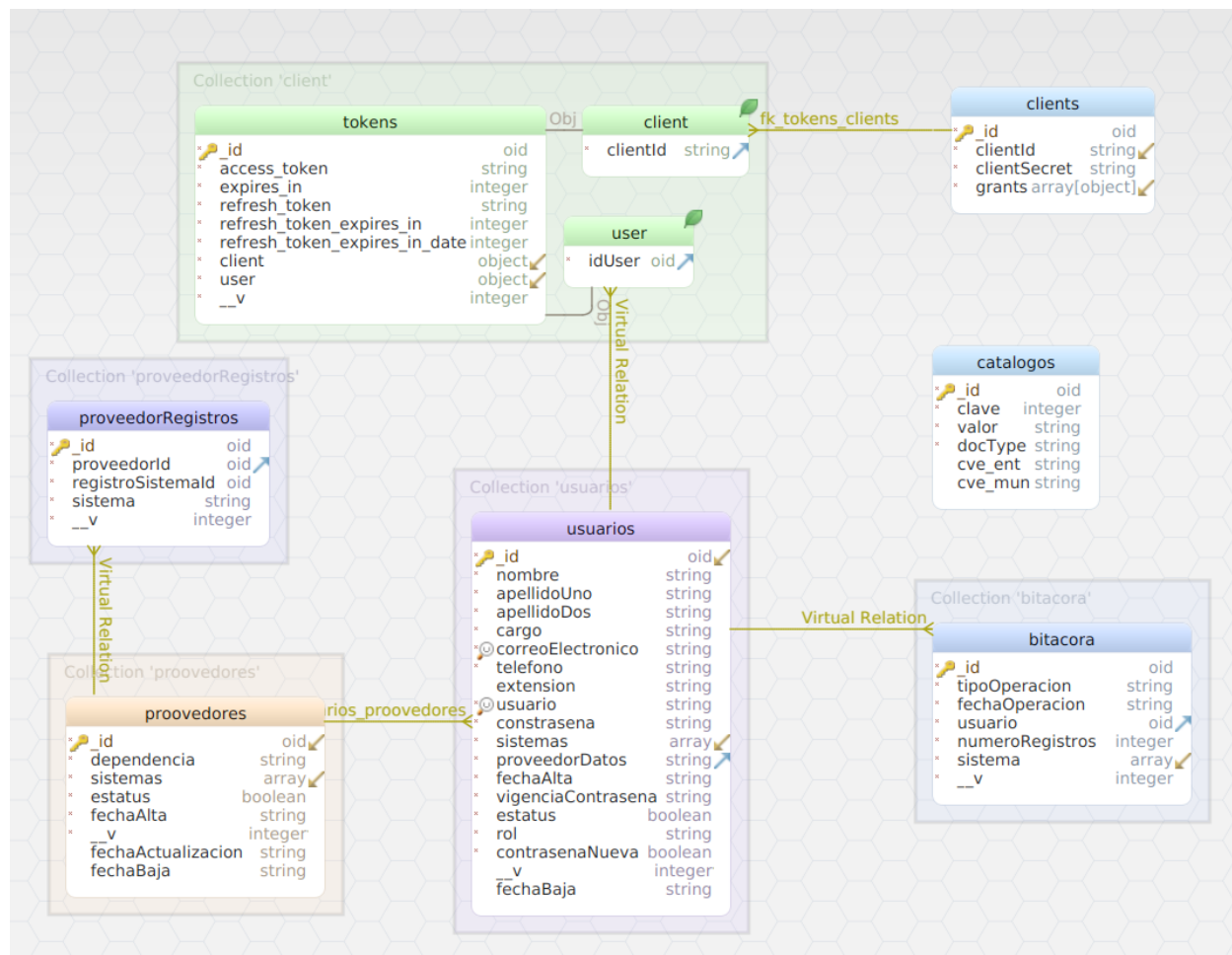
El sistema está basado en una arquitectura cliente - servidor. El cliente o frontend está desarrollado en React JS, por su parte el servidor o backend está desarrollado en Node.js. El frontend se comunica con el backend, quien utiliza un servidor de autorizaciones para la parte de administración de usuarios y permisos, así como una base de datos en MongoDB para el almacenamiento y consulta de información.

Todo lo anterior reside en contenedores Docker.



3.2. Modelado

3.3. Modelo de la base de datos



3.4. Esquema usuarios

Almacena los usuarios del sistema, permitiendo controlar aspectos de seguridad, como el usuario, contraseña, estatus y vigencia.

Path	Campo	Tipo	Descripción
/	id	String	Identificador del registro
/	nombre	String	Nombre del usuario
/	apellidoUno	String	Primer apellido del usuario
/	apellidoDos	String	Segundo apellido del usuario
/	cargo	String	Cargo o puesto del usuario
/	correoElectronico	String	Dirección de correo electrónico
/	telefono	String	Número telefónico
/	extension	String	Número de extensión telefónica
/	usuario	String	Identificador del usuario
/	contrasena	String	Contraseña del usuario
/	sistemas	Array(String)	Sistemas permitidos para el usuario. Ejm: ["S2" ," S3S" ," S3P"]
/	fechaAlta	String	Fecha de alta del usuario
/	fechaBaja	String	Fecha de baja del usuario
/	estatus	Boolean	Estatus del usuario
/	vigenciaContraseña	String	Vigencia de la contraseña del usuario

/	contrasenaNueva	String	Verifica si la contraseña es nueva, siendo así(true) te pedirá cambiarla
/	rol	String	Rol del usuario. Ejm: “1” (Administrador), “2” (Usuario)

3.5. Esquema proveedores

Almacena los proveedores de datos que tendrá la dependencia o ente, así como los sistemas para los que el proveedor entregará información.

Path	Campo	Tipo	Descripción
/	_id	String	Identificador del registro
/	dependencia	String	Nombre de la institución o dependencia proveedora de la información
/	sistemas	Array(String)	Sistemas alimentados por la dependencia. Ejm: [“S2” ,” S3S” ,” S3P”]
/	fechaAlta	String	Fecha de alta del proveedor
/	fechaBaja	String	Fecha de baja del proveedor.
/	fechaActualizacion	String	Fecha de la última actualización de la dependencia
/	estatus	Boolean	Estatus de la dependencia. True (Vigente), False (No vigente)

3.6. Esquema catalogos

Se almacenan todos los catálogos utilizados en el sistema, su estructura general es clave y valor, pero se identifican por el campo docType.

Path	Campo	Tipo	Descripción
/	_id	String	Identificador del registro
/	clave	String	Clave del valor del catálogo
/	valor	String	Valor del catálogo
/	docType	String	Tipo/Nombre del catálogo

3.7. Esquema proveedorRegistros

Almacena la relación entre el proveedor y los registros que le pertenecen, de esta manera se garantiza que únicamente los usuarios del proveedor tengan acceso a sus datos.

Path	Campo	Tipo	Descripción
/	_id	String	Identificador del registro
/	proveedorId	ObjectId	ID del proveedor al que pertenece el registro
/	registroSistemaId	ObjectId	ID del registro
/	sistema	String	Sistema al que pertenece el registro

3.8. Esquema bitacora

Almacena un registro de las operaciones llevadas a cabo en el sistema, permitiendo contar con información del usuario que lleva a cabo la operación, el tipo de operación que realizó y el número de registros afectados.

Path	Campo	Tipo	Descripción
/	_id	String	Identificador del registro
/	tipoOperacion	String	Tipo de operación realizada
/	fechaOperacion	String	Fecha de la operación
/	usuario	ObjectId	Identificador del usuario quien realizó la operación
/	numeroRegistros	number	Número de registros afectados
/	sistema	String	Sistema al que se le realizó la operación

4. Configuración del ambiente

4.1. Instalación de Docker

En caso de haber implementado la fase 1 del proyecto se asume que se tiene docker , docker-compose y MongoDB instalados. Dirigirse a la sección: [Creación de Usuarios](#)

Para entrar a la consola de Linux CentOS dentro del ambiente gráfico de Linux ejecuta el siguiente metacomando CTRL + ALT + T.

Nota: Si la pantalla muestra un signo de dólar (\$) o un hash (#) a la izquierda del cursor parpadeante, está en el entorno de la línea de comandos. Los símbolos \$, #, indican el tipo de cuenta de usuario en el que ha iniciado sesión. El signo de dólar (\$) significa que usted es un usuario normal y el signo de hash (#) significa que usted es el administrador del sistema (root).

Los pasos para instalar Docker mostrados en este documento, se basan en el requerimiento de tener instalada la versión de CentOS 7. En caso de utilizar otra distribución se podrán presentar diferencias en los resultados de las ejecuciones de los comandos utilizados aquí.

1. En caso de que el equipo en el que se realice el proceso de configuración no sea una instalación nueva y puedas tener versiones anteriores de Docker se recomienda desinstalarlas.

Para desinstalar las versiones anteriores, ejecutar el siguiente comando:

```
$ sudo yum remove -y docker \
    docker-client \
    docker-client-latest \
    docker-common \
    docker-latest \
    docker-latest-logrotate \
    docker-logrotate \
```

```
docker-engine
```

2. Actualizar e instalar **yum-utils** y configurar el repositorio

```
$ sudo yum update -y  
  
$ sudo yum install -y yum-utils  
  
$ sudo yum-config-manager \  
    --add-repo \  
    https://download.docker.com/linux/centos/docker-ce.repo
```

3. Instalar la última versión del motor de Docker.

```
$ sudo yum install -y docker-ce docker-ce-cli containerd.io
```

Si se le solicita que acepte la clave GPG, verifique que la huella digital (fingerprint) coincida con la siguiente:

060A 61C5 1B55 8A7F 742B 77AA C52F EB6B 621E 9F35

Si es así, aceptar. En caso contrario, contactar al equipo de soporte.

4. Habilitar el servicio automáticamente cuando se encienda el equipo e Inicializar Docker

```
$ sudo systemctl enable --now docker
```

5. Verificar que Docker esté instalado correctamente y corriendo.

```
$ sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
0e03bdcc26d7: Pull complete
Digest:
sha256:49a1c8800c94df04e9658809b006fd8a686cab8028d33cfba2cc049724254202
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs
    the executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.
To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/
```

6. Agregar tu usuario *<your-user>* al grupo docker con el siguiente comando:

```
$ sudo usermod -aG docker $USER
```

Nota: Deberá cerrar la sesión y volver a iniciarla para que este cambio surta efecto. También puedes abrir una nueva ventana de terminal y continuar en ella los siguientes pasos.

Información detallada de la instalación de Docker la puede encontrar en la siguiente dirección <https://docs.docker.com/engine/install/centos/>

4.2. Instalación de Docker compose

Para la instalación de Docker compose, sigue los siguientes pasos:

1. Descargar la versión más reciente estable de docker compose.

```
$ sudo curl -L  
"https://github.com/docker/compose/releases/download/1.26.2/docker-compose-  
$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

2. Aplicar permisos de ejecución al archivo binario.

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

3. Probar instalación de docker-compose

```
$ docker-compose --version  
  
docker-compose version 1.26.2, build 1110ad01
```

Información detallada de la instalación de docker compose la puede encontrar en la siguiente dirección <https://docs.docker.com/compose/install/>

4.3. Instalación de git

Para poder clonar los repositorios del proyecto, instala git.

```
$ sudo yum install -y git wget vim
```

4.4. Instalación de MongoDB

Antes de comenzar la instalación de MongoDB, verifica que el servicio de Docker esté corriendo:

```
$ sudo systemctl docker status
```

Nota: Si la pantalla muestra un signo de dólar (\$) o un hash (#) a la izquierda del cursor parpadeante, está en el entorno de la línea de comandos. Los símbolos \$, #, indican el tipo de cuenta de usuario en el que ha iniciado sesión. El signo de dólar (\$) significa que usted es un usuario normal y el signo de hash (#) significa que usted es el administrador del sistema (root).

La siguiente imagen muestra el servicio de Docker activo y corriendo:


```
[phoenixnap@localhost ~]$ sudo service docker status
Redirecting to /bin/systemctl status docker.service
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Tue 2020-01-21 05:47:58 MST; 1min 33s ago
     Docs: http://docs.docker.com
    Main PID: 3466 (dockerd-current)
      Tasks: 19
    CGroup: /system.slice/docker.service
            └─3466 /usr/bin/dockerd-current --add-runtime docker-runc=/usr/libexec/do...
              3528 /usr/bin/docker-containerd-current -l unix:///var/run/docker/libco...

Jan 21 05:47:52 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:52.123456789Z" level=info msg="Starting dockerd"
Jan 21 05:47:53 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:53.123456789Z" level=info msg="Listening for connections"
Jan 21 05:47:56 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:56.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:56 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:56.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:57 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:57.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:58 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:58.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:58 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:58.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:58 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:58.123456789Z" level=info msg="Starting daemon"
Jan 21 05:47:58 localhost.localdomain systemd[1]: Started Docker Application Container Engine.
Jan 21 05:47:58 localhost.localdomain dockerd-current[3466]: time="2020-01-21T05:47:58.123456789Z" level=info msg="Starting daemon"
Hint: Some lines were ellipsized, use -l to show in full.
```

4.4.1. Configuración de Docker Compose para MongoDB

A continuación, se configurará MongoDB para trabajar en un contenedor:

1. Crear el directorio **sistema_pdn** dentro del *<home directory>*, dentro se debe crear el subdirectorio **mongodb** y dentro de él se crearán los directorios **volume** y **log**.

```
$ cd
$ mkdir -p sistema_pdn/mongodb/{volume,log}
```

Nota: Dentro de este documento, el directorio <base path> se refiere al directorio <home directory>/sistema_pdn

2. Crear el subdirectorio **log** nos permite tener un registro de los eventos y es útil para la resolución de problemas relacionados con la base de datos. Ahora asignaremos los permisos necesarios

```
$ cd  
$ sudo chmod 777 -R sistema_pdn/mongodb
```

3. Crear el archivo **docker-compose.yml** dentro de la carpeta **sistema_pdn** en caso de que no exista.

La estructura del archivo está en forma de árbol, debajo de la sección de services se puede ver la configuración del contenedor mongodb, en donde **image:mongo** es la imagen base a utilizar para su creación, **restart:always** indica que el contenedor debe reiniciarse cada vez que este se detiene. En caso de detenerse manualmente, este se reinicia solo cuando el Docker daemon o proceso se reinicia o cuando el contenedor se reinicia manualmente.

En este archivo se definen aspectos importantes como son: la imagen base **mongo**, el puerto default por el que escucha: **27017**, las variables de ambiente **MONGO_INITDB_ROOT_USERNAME** y **MONGO_INITDB_ROOT_PASSWORD** que define el usuario root, el path del volumen de datos **./mongodb/volume/**, el path del archivo de configuración de mongo **./mongodb/mongod.conf**, el path del log **./mongodb/log/**, y el archivo **.env** donde se definen las variables de ambiente.

A continuación se muestra el contenido del archivo **docker-compose.yml** referente a la configuración de la base de datos de MongoDB.

```
version: '3.1'  
services:  
  mongodb:  
    image: mongo  
    container_name: mongodb  
    restart: always  
    ports:
```

```
- 27017:27017
environment:
  MONGO_INITDB_ROOT_USERNAME: ${DB_ROOT_USER}
  MONGO_INITDB_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
volumes:
  - ./mongodb/mongo-volume:/data/db
  - ./mongodb/mongod.conf:/etc/mongod.conf
  - ./mongodb/log/:/var/log/mongodb/
env_file:
  - .env
command: ["-f", "/etc/mongod.conf"]
```

4.4.2. Archivo de configuración de la base de datos (mongod.conf)

Para una configuración de la base de datos de MongoDB, crear el archivo de configuración **`mongod.conf`** dentro del directorio de **`mongodb`** con el siguiente contenido, ten en cuenta que la indentación es importante:

```
# mongod.conf
# for documentation of all options, see:
#   http://docs.mongodb.org/manual/reference/configuration-options/
# Where and how to store data.
storage:
  dbPath: /data/db
  journal:
    enabled: true

# where to write logging data.
systemLog:
  destination: file
  logAppend: true
  path: /var/log/mongodb/mongod.log
# network interfaces
net:
  port: 27017
  bindIp: 127.0.0.1
# how the process runs
```

```
processManagement:
  timeZoneInfo: /usr/share/zoneinfo
security:
  authorization: enabled
```

A continuación, se presenta una descripción de los parámetros de configuración:

- **bindIp:127.0.0.1**, el cual forzará que el servidor solo escuche peticiones de la IP de localhost. Solo haga bind a interfaces seguras a las que los sistemas de nivel de aplicación puedan acceder con el control de acceso proporcionado por el filtrado de la red del sistema (es decir, "firewall").
- **port: 27017**, el puerto default para MongoDB, sin embargo se puede especificar cualquier puerto, también puede filtrar el acceso según el puerto utilizando herramientas de filtrado de red.
- **dbPath:/data/db**, especifica el directorio donde se guardaran los archivos de datos de MongoDB. La cuenta que corre mongod necesitará acceso de lectura y escritura a este directorio.

La referencia de este archivo igualmente deberá de montarse en un volumen en el host o también llamada máquina huésped y especificado dentro del archivo **docker-compose.yml**, esto permitirá persistir los datos almacenados en MongoDB aún y cuando el contenedor se apague o deje de funcionar, esto aunado a los respaldos hechos periódicamente permite protegerse de cualquier contingencia que involucre la pérdida de datos.

- **systemLog.path: /var/log/mongodb/mongod.log** es el path del log de mongo.

Al igual que el archivo de datos deberá de montarse en un volumen en el host o máquina huésped y ser especificado dentro del archivo **docker-compose.yml**.

El contar con este tipo de archivos permite tener información sobre el funcionamiento de la base de datos, así mismo ayuda a dar seguimiento de problemas relacionados mediante la inspección de este archivo dinámico.

Para revisar el log de la base de datos de MongoDB se puede usar el siguiente comando que hace referencia al path donde se encuentre mongod.log que configuró previamente:

```
$ sudo tail -f <base path>/mongodb/log/mongod.log
```

- logAppend: **true**, garantiza que mongod no sobrescriba el log después de una operación de inicio de servidor.
- destination: **file**, el destino al que MongoDB envía toda la salida del registro. Especifique archivo o syslog. Si especifica el archivo, también debe especificar systemLog.path.
- storage.journal.enabled :**true**, la cual habilita la opción de journaling o diario. Los cambios en los archivos mapeados en memoria no se aplican en orden y diferentes partes de un archivo pueden ser de diferentes momentos, si algo sucede y MongoDB se cierra inesperadamente queremos volver a tener un estado consistente.
- authorization: **enabled**, los clientes que se conectan a esta instancia ahora deben autenticarse como usuarios de MongoDB. Los clientes solo pueden realizar acciones según lo determinen sus roles asignados.
- timeZoneInfo: **/usr/share/zoneinfo**, la ruta completa desde la que cargar la base de datos de zona horaria. Si no se proporciona esta opción, MongoDB utilizará su base de datos de zona horaria integrada.

Existe un número de opciones referentes a systemLog para el control del comportamiento de los archivos log de la base de datos, para mayor información, vea <https://docs.mongodb.com/manual/reference/configuration-options/#systemlog-options>

Dada la configuración default algunos de estos valores pueden resultar redundantes, en muchas situaciones especificar la configuración incrementa la inteligibilidad global del sistema. La lista completa de opciones del archivo de configuración la puede encontrar en la ruta: <https://docs.mongodb.com/manual/reference/configuration-options/>

4.4.3. Archivo de configuración .env

El archivo **.env** contiene la definición de variables de entorno a utilizar dentro del mismo archivo docker-compose.yml. A continuación se muestra el contenido del archivo .env:

```
# MongoDB
DB_ROOT_USER=root
DB_ROOT_PASSWORD=<password de root>
```

Este archivo deberá ser creado dentro del directorio **sistema_pdn** junto al archivo **docker-compose.yml**.

Los archivos .env ayudan a definir y organizar mejor las variables de entorno ubicándose en archivos especiales, esto permite poder hacer referencia a ellas en archivos de configuración, evitar modificar múltiples archivos y minimizar errores.

4.4.4. Despliegue de la base de datos en Docker Compose

1. Ejecutar el siguiente comando desde el directorio **mongodb** en donde se encuentra el archivo **docker-compose.yml**.

```
$ docker-compose up -d
```

2. Para asegurar que los componentes hayan sido desplegados correctamente, ejecute corra el siguiente comando.

```
$ docker-compose ps
```

Name	Command	State	Ports

mongodb	docker-entrypoint.sh -f /e ...	Up	0.0.0.0:27017->27017/tcp

Nota: el resultado puede variar de la imagen mostrada, verifique que los servicios definidos estén presentes en el resultado y corriendo con State = Up.

4.4.5. Pruebas de operación

Para verificar que la base de datos está funcionando correctamente, se entrará a la Shell de MongoDB:

1. Ejecutar el siguiente comando desde el directorio **sistema_pdn** en donde se encuentra el archivo **docker-compose.yml**.

```
$ docker exec -it mongodb bash
```

2. Ingresar al gestor de base de datos de MongoDB:

```
# mongo -u root -p $MONGO_INITDB_ROOT_PASSWORD
```

3. Se desplegará el prompt de mongo “>” podemos confirmar que funciona.

4.5. Creación de usuarios

4.5.1. Creación de usuarios

Los usuarios creados en MongoDB están asociados a una base de datos, tienen un nombre de usuario, un password y un rol. A continuación se muestra un diagrama que muestra las bases de datos y los usuarios que se tienen que crear.

Base de Datos	Descripción
administracionUsuarios	Almacena la información del sistema respecto a usuarios y bitácora

Usuario	Descripción	Roles
admonP2	Usuario para gestión del sistema fase 2	Read Write Bases de Datos: [administracionUsuarios]

Para crear al usuario seguir los siguientes pasos:

1. Firmarse en la terminal de MongoDB (Ver 3.4.5) y ejecutar los siguientes comandos

```
# mongo -u root -p $MONGO_INITDB_ROOT_PASSWORD

> use administracionUsuarios

> db.createUser({
  "user": "admonP2",
  "pwd": "<password>",
  "roles": [
    {
      "role": "readWrite",
      "db": "administracionUsuarios"
    }
  ]
});
```

Se recomienda que la longitud de las contraseñas (<password>) sea de 8 a 10 caracteres alfanuméricos combinados, altas, bajas y algunos caracteres especiales intercalados como “_” ,” @” ,” \$” ,” #” ,” /” ,” =” , documento y almacene usuarios y passwords de forma segura.

2. Verificar que el usuario se haya creado correctamente:

```
> use administracionUsuarios
> db.getUsers()

[
  {
    "_id" : "administracionUsuarios.admonP2",
    "userId" : UUID("c4a08711-330a-40fd-9b58-f6592a581dff"),
    "user" : "admonP2",
    "db" : "administracionUsuarios",
    "roles" : [
      {
        "role" : "readWrite",
        "db" : "administracionUsuarios"
      }
    ],
    "mechanisms" : [
      "SCRAM-SHA-1",
      "SCRAM-SHA-256"
    ]
  },
]
```

3. Ejecutar la función **grantRolesToUser** para otorgar permisos al usuario **admonP2** de conectarse a las bases de datos **S2**, **S3_Servidores** y **S3_Particulares**:

```
> db.grantRolesToUser( "admonP2", [ { role: "readWrite", db: "S2" } ])
> db.grantRolesToUser( "admonP2", [ { role: "readWrite", db: "S3_Servidores" } ])
> db.grantRolesToUser( "admonP2", [ { role: "readWrite", db: "S3_Particulares" } ]])
```

4.5.2. Creación de usuario administrador

Para el uso del sistema, se requiere tener un usuario administrador inicial. Para crearlo, en la consola de Linux se deben ejecutar los siguientes comandos:

1. Entra al Shell de MongoDB (Ver 3.4.5)
2. Ejecuta:

```
> use administracionUsuarios
> db.usuarios.insertOne({
  "nombre": "example",
  "apellidoUno": "example",
  "apellidoDos": "example",
  "cargo": "Administrador",
  "correoElectronico": "admin@gmail.com",
  "telefono": "0000000000",
  "extension": "626262",
  "usuario": "exampleAdmon",
  "contrasena": "87dsa.3j",
  "sistemas": [
    "S2",
    "S3S",
    "S3P"
  ],
  "proveedorDatos": "iodsioadsijsdaijosda",
  "fechaAlta": "2021-01-15T15:28:28-06:00",
  "vigenciaContrasena": "2099-04-15T15:28:28-05:00",
  "estatus": true,
  "rol": "1",
  "contrasenaNueva": false
})

> db.clients.insert([{
  clientId: 'txm.global',
  clientSecret: 'pass',
  grants: []
}
])
```

3. Verificar los registros insertados con los siguientes comandos:

```
> db.usuarios.find().pretty()

> db.clients.find().pretty()
```

4. Bajo esta configuración, las credenciales del usuario administrador para iniciar sesión son:

Usuario	exampleAdmon
Contraseña	87dsa.3j

4.6. Creación y carga de los catálogos

El sistema requiere ciertos catálogos para la captura de los datos de los diferentes sistemas, en la siguiente tabla se muestran los nombres de los catálogos y el nombre del archivo donde se encuentra la información:

Nombre catálogo	Nombre archivo
Categorías/Ramos	categoriaRamo.json
Estados	estados.json
Géneros	generos.json
Localidades	Localidades.json
Monedas	monedas.json
Municipios	municipios.json

Niveles de Responsabilidad	nivelResponsabilidad.json
País	pais.json
Tipos de Área	tipoArea.json
Tipos de Documento	tipoDocumento.json
Tipos de Falta	tipoFalta.json
Tipos de Persona	tipoPersona.json
Tipos de Procedimiento	tipoProcedimiento.json
Tipo de Sanción S3S	tipoSancion.json
Tipo de Sanción S3P	tipoSancionS3P.json
Vialidades	vialidades.json

Seguir los siguientes pasos para la carga de catálogos:

1. Descarga los archivos en **<base path>**

```
$ git clone https://github.com/PDNMX/piloto_sistema_catalogos.git
```

2. Busca el ID del contenedor de mongo. Ejemplo:

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
0e4cb498d26f	mongo	"docker-entrypoint.s..."	9 hours ago	Up 9 hours
0.0.0.0:27017->27017/tcp		mongodb		

3. Reemplaza el ID en el siguiente comando:

```
$ sudo docker cp <base path>/piloto_sistema_catalogos <container_id>:/home
```

4. Entra a la Shell de MongoDB

```
$ docker exec -it <container_id> bash
```

Dentro del contenedor

```
# cd /home/piloto_sistema_catalogos/
```

5. Sustituye el password del usuario **admonP2** en el siguiente comando, a continuación se insertarán los registros de cada uno de los archivos que existen dentro de la carpeta:

```
$ for file in $(ls *.json);
do
    echo $file;
    mongoimport -d administracionUsuarios -c catalogos --jsonArray -u
admonP2 -p <password> --type json --file $file
done
```

- Verifica obtener como resultado el número de documentos importados y el mensaje de que el comando fue exitoso de cada uno de los archivos. Ejemplo:

```
Catalogo_Municipios.json
2021-04-08T15:30:44.658+0000    connected to: mongodb://192.168.56.5:27017/
2021-04-08T15:30:44.694+0000    762 document(s) imported successfully. 0 document(s) failed to import.
Catalogo_estado.json
2021-04-08T15:30:44.736+0000    connected to: mongodb://192.168.56.5:27017/
2021-04-08T15:30:44.738+0000    32 document(s) imported successfully. 0 document(s) failed to import.
Catalogo_vialidad.json
```

- Salimos del contenedor

```
# exit
```

4.7. Instalación del servidor OAuth2

El servidor OAuth se encuentra en el repositorio de Github de la PDN en el siguiente link:

https://github.com/PDNMX/piloto_sistema_oauth20.git. Para su instalación sigue los siguientes pasos:

- Ejecutar el comando de clonado desde la ruta **<base path>** con la ayuda del CLI (Interfaz de línea de comandos) de su computador, utilice el siguiente comando:

```
$ git clone https://github.com/PDNMX/piloto_sistema_oauth20.git
```

- Abrir el archivo .env

```
$ cd piloto_sistema_oauth20
$ vi config/config.env
```

3. Fijar las variables de entorno dependiendo del mismo:

- **USERMONGO:** Contiene el nombre del usuario **admonP2** para acceder a la base de datos **administracionUsuarios**. Este usuario debe contar con los privilegios correspondientes y fue creado durante la instalación de MongoDB.
- **PASSWORDMONGO:** Contiene la contraseña *<password>* del usuario **admonP2** para acceder a la base de datos **administracionUsuarios**.
- **HOSTMONGO:** Contiene la dirección IP mongodb y el puerto 27017 donde se encuentra alojada la base de datos. Se utiliza el link mongodb que se encuentra declarado en el archivo `docker-compose.yml` en lugar de la dirección IP. En este caso, tiene el valor de `mongodb:27017`.
- **DATABASE:** Contiene el nombre de la base de datos **administracionUsuarios**.
- **PORTSERVER:** Contiene el puerto 9004 en el que se ejecuta la aplicación.

Las variables de entorno que hacen referencia a nuestros tokens

- **EXT:** Contiene el número de segundos 600 para la expiración del token.
- **RTEXT:** Contiene el número de segundos 900 para la expiración del refresh token. Este número debe ser mayor al tiempo de expiración del token.
- **SEED:** Contiene la semilla `YTBGD9YjAUhkjQk9ZXcb` para la codificación del JSON Web Token (JWT)
- Cabe mencionar que el refresh token es una cadena aleatoria no un JWT, ya que el refresh token se valida en el servidor de autorización. Por lo tanto, se

puede obtener haciendo una petición a la base de datos y verificando los parámetros para generar un nuevo token.

```
#Expiration token in seconds
EXT=600

#Expiration refresh token in seconds
RTEXT=900

#SEED
SEED=YTBGD9YjAUhkjQk9ZXcb

#MONGO CONFIG

USERMONGO=admonP2
PASSWORDMONGO=<password>
HOSTMONGO=mongodb:27017
DATABASE=administracionUsuarios
PORTSERVER=9004
```

4. Agrega el servicio oauth20v2 al archivo docker-compose.yml que se encuentra en el directorio **<home directory>/sistema_pdn** de la siguiente forma, teniendo en cuenta que la indentación es muy importante:

```
version: '3.1'
services:
  ....
  oauth20v2:
    restart: always
    container_name: oauth20v2
    build:
      context: piloto_sistema_oauth20
      dockerfile: Dockerfile
    ports:
      - 9004:9004
    links:
      - mongodb
    depends_on:
      - mongodb
  ...
```

5. Despliega el contenedor, ejecutando lo siguiente desde el directorio **<home directory>/sistema_pdn** que contiene el archivo docker-compose.yml

```
$ docker-compose build oauth20v2
$ docker-compose up -d oauth20v2
```

6. Verificar que todos los contenedores dentro del archivo `docker-compose.yml` estén corriendo con el siguiente comando. Ten en cuenta que el resultado de la imagen puede variar:

```
$ docker-compose ps
```

Name	Command	State	Ports
mongodb	docker-entrypoint.sh -f /e ...	Up	0.0.0.0:27017->27017/tcp
mongodb_oauth2v2	docker-entrypoint.sh yarn ...	Up	0.0.0.0:9004->9004/tcp

4.7.1. Esquema de la base de datos

Dentro de MongoDB, se tiene la base de datos **administracionUsuarios** para las siguientes colecciones que conciernen a la implementación OAuth 2.0, y que ya fueron creadas durante el proceso de instalación de MongoDB.

4.7.1.1. Usuarios

Nombre de la colección: **usuarios**

Campo	Tipo	Valor ejemplo	Descripción
usuario	String	'jperez'	Nombre del usuario o identificador
contrasena	String	'123456'	Contraseña del usuario

4.7.1.2. Clientes

Nombre de la colección: **clientes**

Campo	Tipo	Valor ejemplo	Descripción
clientId	String	'pdn'	Identificador del cliente
clientSecret	String	'pass'	Contraseña del cliente
grants	Array String	['admin']	Privilegios asociados con el cliente

Nota: En caso de que el valor clientSecret no se agregue al documento, porque no existe en el ambiente, la solicitud POST no deberá contener dicho parámetro o se puede colocar en el documento con un valor vacío, es decir, dos comillas simples "" .

4.7.1.3. Tokens

Nombre de la colección: **tokens**

Campo	Tipo	Valor ejemplo	Descripción
access_token	String	'eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImVjYW1hcmdvliwiaW1hcnRpljoiriVoQ1dNR0siLCJzY29wZSI6IndyaXRlIiwiaWF0IjE5MTU0MDYyYyQ.n5JLQo0fI7l0LbJ_anPfsA3O2r-EMIGmvK0fJ-LP2Zg'	Token en formato JWT el cual permite la validación de identidad en el recurso protegido

expires_in	Number	300	Tiempo de expiración del access_token en segundos
refresh_token	String	'OztWyf5YGjPJxrkkx4feeRGT3eup21yRMZgfMuNLrwBKvslcKT3u6PvyjGlfc951nLhr0tNOZT4UezG971FXUNBaUNDaWNO6h8Uzno62wJA5K3iRF9smW4IdgmXMpkr4fB0C5KfQmsjNZL02bTzrQBmJ4BEOTmRjseAkr0A3JQU3vFtlyyXHQWxVaW03tNDgu001feEgQ15XilnmWq9zubngnnLLOZrN6bah3UhGxSwFgydgzR9W19CpxDdryrsE'	Token (random string) el cual se utiliza para generar un nuevo token (flujo refresh token)
refresh_token_expires_in	Number	600	Tiempo de expiración del refresh token en segundos
refresh_token_expires_in_date	Number	1595275826	unix timestamp de la expiración del refresh token
client	Object	{clientId: 'pdn.resource.1'}	Se almacena solo el ID del cliente
user	Object	{idUser: '5fdd06611cd073253082a15e'}	Se almacena el idUser

4.7.2. Pruebas de operación

Para poder generar un nuevo token se requiere enviar una solicitud de tipo POST a la ruta `http://<IP_HOST>:9004/oauth/token`. La variable `<IP_HOST>` es la dirección IP donde está instalado el Sistema.

Ten en cuenta que el parámetro **client_secret** es opcional, puede ir o no dentro del documento de la base de datos **administracionUsuarios** y por tanto, puede omitirse de las peticiones para generar el token.

Dicha petición se puede estructurar de dos maneras diferentes:

4.7.2.1. Generar un token usando base 64

1. Generar el formato base 64 utilizando los valores de **client_id** y **client_secret** en los campos **username** y **password** del siguiente enlace
<https://www.blitter.se/utils/basic-authentication-header-generator/>
2. Obtener el token utilizando el formato base64 generado en el paso anterior, por ejemplo, `dHhtLmdsb2JhbDpwYXNz`, y ejecutando el siguiente comando desde el CLI del sistema operativo de nuestro ordenador:

```
curl --location --request POST '<IP_HOST>:9004/oauth/token' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--header 'Authorization: Basic dHhtLmdsb2JhbDpwYXNz' \  
--data-urlencode 'grant_type=password' \  
--data-urlencode 'username=exampleAdmon' \  
--data-urlencode 'password=87dsa.3j'
```

4.7.2.2. Generar un token a través del body

1. Ejecuta el siguiente comando desde el CLI del sistema operativo de tu ordenador:

```
curl --location --request POST '<IP_HOST>:9004/oauth/token' \  
--header 'Content-Type: application/x-www-form-urlencoded' \  
--data-urlencode 'client_id=pdn' \  
--data-urlencode 'grant_type=password' \  
--data-urlencode 'username=exampleAdmon' \  
--data-urlencode 'password=87dsa.3j' \  
--data-urlencode 'scope=' \  
--data-urlencode 'client_secret=pass'
```

4.7.2.3. Respuesta

La respuesta esperada en ambos métodos de obtención de token es similar a la que se muestra a continuación:

```
{  
  "access_token": <cadena>,  
  "token_type": "bearer",  
  "expires_in": 300,  
  "refresh_token": <cadena>,  
  "refresh_token_expires_in": 600,  
  "scope": "read"  
}
```

4.8. Instalación del Backend

El backend se encuentra en el repositorio de GitHub de la PDN en el siguiente link:

https://github.com/PDNMX/piloto_sistema_backend.git. Para su instalación sigue los siguientes pasos:

1. Ejecutar el comando de clonado desde la ruta *<base path>* con la ayuda del CLI (Interfaz de línea de comandos) de su computador, utilice el siguiente comando:

```
$ git clone https://github.com/PDNMX/piloto_sistema_backend.git
```

2. Abrir el archivo .env

```
$ cd piloto_sistema_backend  
$ vi .env
```

3. Fijar las variables de entorno dependiendo del mismo:
 - **USERMONGO:** Contiene el nombre del usuario admonP2 para acceder a la base de datos **administracionUsuarios**. Este usuario debe contar con los privilegios correspondientes y fue creado durante la instalación de MongoDB.
 - **PASSWORDMONGO:** Contiene la contraseña *<password>* del usuario admonP2 para acceder a la base de datos **administracionUsuarios**.
 - **HOSTMONGO:** Contiene la dirección IP_HOST y el puerto 27017 donde se encuentra alojada la base de datos.
 - **DATABASE:** Contiene el nombre de la base de datos **administracionUsuarios**.

- **SEED:** Contiene la semilla YTBGD9YjAUhkjQk9ZXcb para la codificación del JSON Web Token (JWT)
- **SEED debe contener el mismo valor que se colocó en las variables de entorno del sistema de autorización Oauth 2.0**
- **EMAIL:** Es el correo electrónico que servirá para enviar cuando se registra un usuario o se restablece una contraseña
- **PASS_EMAIL:** Contraseña del correo electrónico
- **HOST_EMAIL:** Servidor SMTP del correo por ejemplo smtp.google.com

```
USERMONGO=admonP2
PASSWORDMONGO=<password>
HOSTMONGO=<IP_HOST>:27017
DATABASE=administracionUsuarios
SEED=YTBGD9YjAUhkjQk9ZXcb

EMAIL=<correo_electronico>
PASS_EMAIL=<password_correo_electronico>
HOST_EMAIL=<servidor smtp>
```

4.8.1. Configuración de Docker Compose para backend

Ejecuta los siguientes pasos:

1. Edita el archivo **docker-compose.yml** que se encuentra dentro de la carpeta sistema_pdn

```
$ cd <base path>/sistema_pdn
$ vi docker-compose.yml
```

2. Agrega la configuración correspondiente al backend. La indentación es importante.

```
version: '3.1'

services:
  mongodb:
    image: mongo
    ...

    command: ["-f", "/etc/mongod.conf"]
  backend:
    restart: always
    container_name: backend
    build:
      context: piloto_sistema_backend
      dockerfile: Dockerfile
    ports:
      - 3004:3004
```

3. Compila la imagen dentro de la misma carpeta <base path>/mongodb

```
$ docker-compose build backend
```

4. Despliega

```
$ docker-compose up -d
```

Nota: Siempre que se modifique el código del proyecto ya sea variables de entorno, actualización del repositorio etc. deberá de ejecutarse ambos pasos, es decir el “build” y el “up -d”

5. Verifica que los componentes hayan sido desplegados correctamente. El resultado de la imagen puede variar, sin embargo, asegúrate que el servicio backend está presente.

```
$ docker-compose ps
```

Name	Command	State
Ports		

backend	docker-entrypoint.sh npm start	Up
0.0.0.0:3004->3004/tcp		
...		

4.8.2. Pruebas de operación

Para garantizar que el backend está funcionando correctamente, ejecuta:

1. Realiza una solicitud CURL al endpoint **/getUsers** (requieren Bearer token, verificar punto “[Generar un token](#)” para obtener uno válido). Ejemplo:

```
curl --location --request POST 'localhost:3004/getUsers' \  
--header 'Authorization: Bearer <EXAMPLETOKEN>'
```

2. En el paso anterior, la respuesta esperada sería, por ejemplo:

```
{
  "pagination": {
    "hasNextPage": true,
    "page": 1,
    "pageSize": 1,
    "totalRows": 1
  },
  "results": [
    {
      "_id": "5fdd06611cd073253082a15e",
      "nombre": "Juan",
      "apellidoUno": "Perez",
      "apellidoDos": "Lopez",
      "cargo": "develloper",
      "correoElectronico": "jlperez@gmail.com",
      "telefono": "4564564654",
      "extension": "456",
      "usuario": "jlperez",
      "constrasena": "123456",
      "sistemas": [
        "s31",
        "S3S",
        "S2"
      ],
      "proveedorDatos": "5fca89136f719226e8e13122",
      "fechaAlta": "2021-01-06T14:34:09-06:00",
      "estatus": true,
      "vigenciaContrasena": "2021-04-06T14:34:09-05:00",
      "__v": 0
    }
  ]
}
```

4.9. Instalación del frontend

El frontend se encuentra en el repositorio de GitHub de la PDN, en el siguiente link:

https://github.com/PDNMX/piloto_sistema_frontend.git. Para su instalación sigue los siguientes pasos:

1. Ejecutar el comando de clonado desde la ruta **<base path>** con la ayuda del CLI (Interfaz de línea de comandos) de su computador, utilice el siguiente comando:

```
$ git clone https://github.com/PDNMX/piloto_sistema_frontend.git
```

2. Abrir el archivo .env

```
$ cd piloto_sistema_frontend  
$ vi .env
```

3. Fijar las variables de entorno dependiendo del mismo:
 - **URLAPI:** Contiene la IP donde está alojada la API backend
 - **PORTOAUTH:** Contiene el puerto en el que se ejecuta el servicio OAuth2 para este proyecto (el 9004 es el configurado en la app de Oauth)
 - **PORTAPI:** Contiene el puerto en el que se ejecuta el servicio del backend para este proyecto (el 3004 es el configurado en la app del backend)
 - **CLIENTID:** Contiene el ID de un cliente ya previamente dado de alta en la base de datos
 - **CLIENTSECRET:** Contiene el password de un cliente ya previamente dado de alta en la base de datos

- **SEED debe contener el mismo valor que se colocó en las variables de entorno del sistema de autorización Oauth 2.0**

```
URLAPI=http://<IP_HOST>  
PORTOAUTH=:9004  
PORTAPI=:3004  
CLIENTID=pdn  
CLIENTSECRET=pass
```

Para los usuarios que estén ejecutando el proyecto en alguna máquina virtual o requiera desplegar en su IP pública el proyecto se requiere hacer el siguiente cambio

```
$ vi <base path>/piloto_sistema_frontend/webpack.config.js
```

Dentro del programa “vim” teclear la letra “i” para entrar en modo de edición del documento y reemplazar el valor de la variable host en el apartado **devServer**, tendrá que quedar de la siguiente manera:

```
...  
devServer: {  
  host: '0.0.0.0',  
  port: 3000,  
  historyApiFallback: true  
},  
...
```

4.9.1. Configuración de Docker Compose para frontend

Ejecuta los siguientes pasos:

1. Edita el archivo **docker-compose.yml** que se encuentra dentro de la carpeta `sistema_pdn`

```
$ cd <base path>/sistema_pdn
$ vi docker-compose.yml
```

2. Agrega la configuración correspondiente al backend. La indentación es importante.

```
version: '3.1'

services:
  mongodb:
    image: mongo
    ...

    command: ["-f", "/etc/mongod.conf"]
  frontend:
    restart: always
    container_name: frontend
    build:
      context: piloto_sistema_frontend
      dockerfile: Dockerfile
    ports:
      - 3000:3000
```

3. Compila la imagen dentro de la misma carpeta <base path>/mongodb

```
$ docker-compose build frontend
```

4. Despliega

```
$ docker-compose up -d
```

Nota: Siempre que se modifique el código del proyecto ya sea variables de entorno, actualización del repositorio etc. deberá de ejecutarse ambos pasos, es decir el “build” y el “up -d”

5. Verifica que los componentes hayan sido desplegados correctamente. El resultado de la imagen puede variar, sin embargo, asegúrate que el servicio frontend está presente.

```
$ docker-compose ps
```

Name	Command	State
Ports		

frontend	docker-entrypoint.sh npm start	Up
0.0.0.0:3000->3000/tcp		
...		

4.9.2. Pruebas de operación

Ingresa al explorador en la dirección pública del servidor o en el localhost en caso de que se quiera ejecutar de manera local en el puerto 3000.

Ejemplo: <http://localhost:3000/login>

Y se desplegará la pantalla correspondiente

Sistema de carga de datos S2 Y S3



Inicio de sesión

Nombre de usuario *

Contraseña *

ENTRAR

[Restablecer contraseña](#)

5. Seguridad

5.1. Firewall

Les recomendamos activen su firewall de CentOS cuando ya tenga correctamente funcionando todo.

Para instalarlo deberán ejecutar el siguiente comando.

```
$ sudo yum install firewalld
```

Para activarlo en cada inicio de sistema y al mismo tiempo iniciarlo

```
$ sudo systemctl enable --now firewalld
```

Para verificar que está corriendo correctamente ejecute

```
$ sudo firewall-cmd --state
```

Deberá configurar el acceso por HTTP y HTTPS, además tendrá que permitir el puerto del backend solo en el caso de que no utilice nginx proxy

```
$ sudo firewall-cmd --zone=public --add-service=ssh
$ sudo firewall-cmd --zone=public --permanent --add-service=ssh

$ sudo firewall-cmd --zone=public --add-service=http
$ sudo firewall-cmd --zone=public --permanent --add-service=http

$ sudo firewall-cmd --zone=public --add-service=https
$ sudo firewall-cmd --zone=public --permanent --add-service=https

$ sudo firewall-cmd --zone=public --add-port=3004/tcp

$ sudo firewall-cmd --reload
```

Cuando termine de ejecutar los siguientes comandos solo podrá acceder al servidor por los puertos de web y por el SSH. De esta manera se genera un ambiente más protegido de sus datos.

5.2. Certificado SSL

SSL (Secure Sockets Layer o capa de conexión segura) es un estándar de seguridad global que permite la transferencia de datos cifrados entre un navegador y un servidor web. Este certificado SSL ofrece seguridad en las peticiones a sistemas de Oauth2, S2, S3_Servidores y S3_Particulares; para esto, se deben realizar las siguientes secciones.

5.2.1. Contar con un dominio

En caso de ya contar con un dominio, omite esta sección. Para obtener un dominio, una posible opción gratuita sería generar el dominio en freenom, para esto se debe realizar los siguientes pasos:

1. Registrar un nuevo dominio en la página <http://www.freenom.com/en/index.html> utilizando la opción **Services** → **register a new domain**
2. Introducir el nombre del dominio a crear en la caja de texto de entrada con el mensaje “**Find a new FREE domain**” y oprimir el botón **Check Availability** para verificar la disponibilidad del nombre del dominio.
3. Seleccionar la terminación del nuevo dominio como se muestra en la siguiente imagen, y posteriormente, oprimir la opción **Checkout**

Check Availability

2 domains in cart [Checkout](#)

Get one of these domains. They are free!

pdnservices .tk	• FREE	USD 0. ⁰⁰	Get it now!
pdnservices .ml	• FREE	USD 0. ⁰⁰	✓ Selected
pdnservices .ga	• FREE	USD 0. ⁰⁰	Get it now!
pdnservices .cf	• FREE	USD 0. ⁰⁰	Get it now!
pdnservices .gq	• FREE	USD 0. ⁰⁰	Get it now!

4. Seleccionar el periodo deseado, y oprimir **Continue** como se muestra en la siguiente imagen

Domain

Use your new domain

pdnservices.ml

Forward this domain
or
 Use DNS

Period

3 Months @ FREE

[Continue](#)

- En la sección siguiente **Review & Checkout** introducir un email válido el cual se asocia a una cuenta después de la validación por correo electrónico

Review & Checkout

Description	Price
Domain Registration - servicepdn1.ml	\$0.00USD
Subtotal:	\$0.00USD
Total Due Today:	\$0.00USD

Please enter your email address and click verify to continue to the next step

Enter Your Email Address

Verify My Email Address

OR

Already Registered? Click here to login

Use social sign in

Sign in

- Entrar al correo mencionado en el paso 5 y verificar en la bandeja de entrada el correo de Freenom dar click en el vínculo de validación.
- Llenar el formulario para el registro de la cuenta, marcar la casilla **“I have read and agree to the Terms & Conditions”** y dar click en **complete orden**.
- Ya se tendrá acceso a la cuenta y en el apartado **My domains** estará el dominio generado.

5.2.2. Vincular la IP del servidor al dominio

Para vincular la IP de un servidor al dominio, se tienen dos opciones según el tipo de servidor, ya sea físico o como instancia en la nube.

Cuando se trate de un servidor físico y se desee vincular su IP pública a un dominio de freenom, se deben de realizar los siguientes pasos:

1. Ir al apartado **MyDomains** de la página <http://www.freenom.com/en/index.html>
2. Oprimir el botón **Manage Domain** del dominio que se quiere vincular a la IP pública
3. Dar click en el menú Management **tools** → **NameServers**
4. Seleccionar la opción **Use default nameservers (Freenom Nameservers)**
5. Ir al apartado **Manage Domain** Seleccionar **Manage Freenom DNS**
6. Agregar un nuevo registro, llenar la información

```
Name : <name>
Type : A
TTL : 3600
Target : <IP_publica>
```

Nota: El valor de **<name>** se deja vacío cuando tú quieres apuntar directamente al dominio raíz (servicepdn.ml). El valor **<IP_publica>** significa la dirección IP pública con la cual se vincula el dominio.

Add Records

Name	Type	TTL	Target
	A	3600	172.0.0.1

⬅ More Records Save Changes

7. Dar clic en **Save Changes**, esperar a que se propague aprox. 30 min

Cuando se trate de un servidor en la nube, y se desee vincular su IP a un dominio de freenom, se deben de realizar los siguientes pasos (Caso Google Cloud):

1. Seleccionar la opción **create zone** en la sección **Cloud DNS** dentro del panel principal de **GCP**
2. Introducir el nombre de la zona, el nombre del dominio y la descripción como se muestra en la siguiente imagen

← Create a DNS zone

A DNS zone is a container of DNS records for the same DNS name suffix. In Cloud DNS, all records in a managed zone are hosted on the same set of Google-operated authoritative name servers. [Learn more](#)

If you don't have a domain yet, purchase one through [Google Domains](#).

Zone type ⓘ

☐ Private

☒ Public

Zone name *
example-pdn ⓘ

DNS name *
servicepdn.ga ⓘ

DNSSEC *
Off ⓘ

Description
example description

After creating your zone, you can add resource record sets and modify the networks your zone is visible on.

CREATE CANCEL

3. Oprimir la opción **ADD RECORD SET** para llenar los campos DNS name (nombre del dominio) y el que refiere a la dirección IP del servidor donde se alberga la aplicación web, en este caso dentro de GCP.

- Configurar los campos Resource Record Type, TTL, TTL Unit, e IPv4 Address como se muestra en la siguiente imagen

DNS Name: .servicepdn.ga.

Resource Record Type: A

TTL *: 5

TTL Unit: minutes

IPv4 Address: 35.226.19.219

- Por último, oprimir la opción create.

5.2.3. Vincular el dominio al servidor

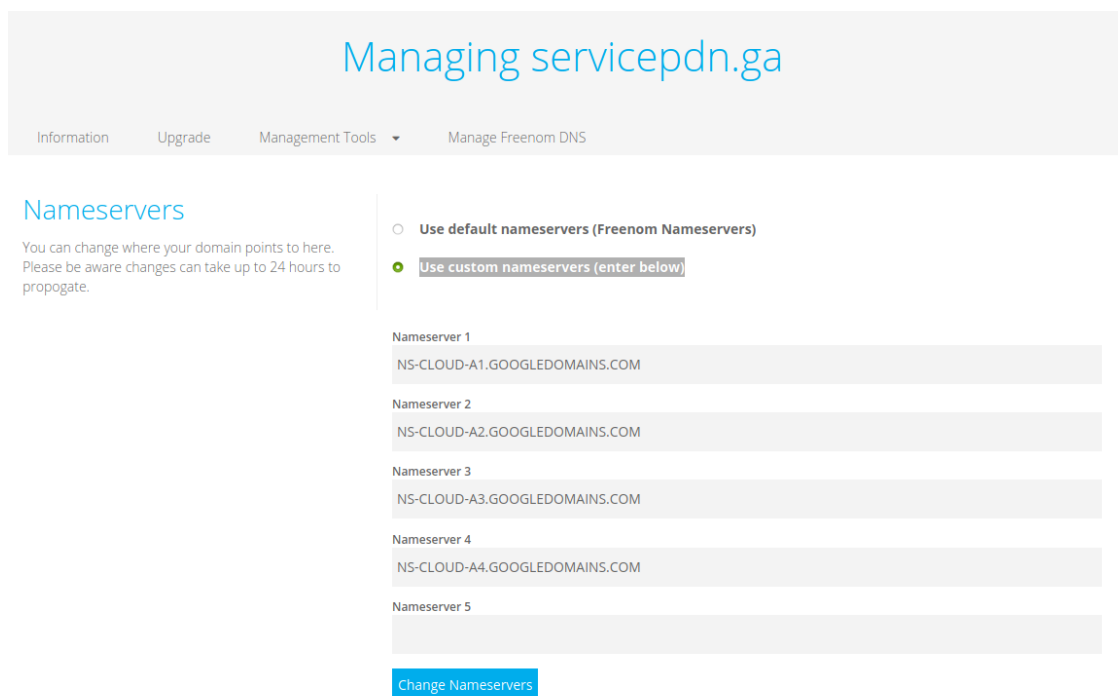
La vinculación del dominio al servidor depende del proveedor del servicio (godaddy, neubox etc.) debido a que los nameservers dependen de la plataforma donde se haya adquirido el servidor. En este caso, se utilizarán los nameservers correspondientes a Google Cloud Platform (GCP). Para verificar que el vínculo sea exitoso en un dominio en freenom, se debe realizar los siguientes pasos:

- Seleccionar la opción del menú Services → MyDomains de la página <http://www.freenom.com/en/index.html>
- Oprimir la opción Manage Domain como se muestra en la siguiente imagen

Domain	Created	Expires	Status	Price	Action
servicepdn.ga	2020-09-04	2021-03-04	ACTIVE	Free	Manage Domain

- Seleccionar la opción Management Tools → name servers

4. Seleccionar la opción Use custom nameservers (enter below)
5. Introducir los DNS de GCP como se muestra en la siguiente imagen



Managing servicepdn.ga

Information Upgrade Management Tools Manage Freenom DNS

Nameservers

You can change where your domain points to here. Please be aware changes can take up to 24 hours to propagate.

☐ Use default nameservers (Freenom Nameservers)
 ☒ Use custom nameservers (enter below)

Nameserver 1
NS-CLOUD-A1.GOOGLEDOMAINS.COM

Nameserver 2
NS-CLOUD-A2.GOOGLEDOMAINS.COM

Nameserver 3
NS-CLOUD-A3.GOOGLEDOMAINS.COM

Nameserver 4
NS-CLOUD-A4.GOOGLEDOMAINS.COM

Nameserver 5

Change Nameservers

6. Oprimir la opción **Change Name servers** y esperar a que se propague la nueva información

Hasta este punto ya se tiene el DNS asociado con el servidor para generar el certificado. Es necesario editar archivo de configuración Webpack del frontend para que el aplicativo permita el acceso a la aplicación con el dominio implementado:

Ruta: **<base path>/piloto_sistema_frontend/**

```
$ vi webpack.config.js
```

Dentro del programa “vi” teclear la letra “i” para entrar en modo de edición del documento y dentro del objeto **devServer** agregar el campo **public** con el valor del dominio generado quedando de la siguiente manera:

```
...
devServer: {
  host: '0.0.0.0',
  port: 3000,
  historyApiFallback: true,
  public: <'servicepdn.ga'>

},
...
```

Nota: se tiene que correr el comando build y el comando up de docker compose para que surta efecto el cambio en la configuración del componente webpack

Editar el archivo env de frontend para que las solicitudes hacia el api y el Oauth se realicen por el protocolo https:

Ruta: **<base path>/piloto_sistema_frontend/**

```
$ vi .env
```

Quedando de la siguiente manera, donde el dominio cambiará dependiendo del generado previamente en la plataforma freenom.

```
URLAPI= https://<servicepdn.ga>  
PORTOAUTH =  
PORTAPI =  
CLIENTID = txm.global  
CLIENTSECRET = pass  
SEED = YTBGD9YjAUhkjQk9ZXcb
```

5.2.4. Implementar Let's Encrypt

Let's Encrypt es una autoridad que tiene la posibilidad de firmar certificados SSL. El procedimiento es generar unos certificados temporales, colocarlos en el servidor y solicitar a Let's Encrypt que los firme. Una vez firmados estos certificados podrá hacer uso de ellos nginx para el cifrado de las conexiones. Para la implementación, siga los siguientes pasos:

1. Agregar al archivo docker-compose.yml el módulo de cerbot junto con algunos volúmenes que usará compartidamente con nginx, también se agrega un apartado para la renovación constante del certificado en el caso de cerbot sea el endpoint.

2. Por el lado de nginx se tienen que agregar los volúmenes y exponer el puerto correspondiente a las conexiones SSL, además de agregar un comando para que se verifique la caducidad del certificado y se autorrenueve. Las secciones mencionadas quedarían de la siguiente manera:

```
nginx:
  image: nginx:latest
  volumes:
    - ./nginx.conf:/etc/nginx/nginx.conf:ro
    - ./data/certbot/conf:/etc/letsencrypt
    - ./data/certbot/www:/var/www/certbot
  depends_on:
    - oauth20v2
    - frontend
    - backend
  ports:
    - "80:80"
    - "443:443"
  command: "/bin/sh -c 'while ;; do sleep 6h & wait $$(!); nginx -s reload;
done & nginx -g \"daemon off;\""
  certbot:
    image: certbot/certbot
    volumes:
      - ./data/certbot/conf:/etc/letsencrypt
      - ./data/certbot/www:/var/www/certbot
    entrypoint: "/bin/sh -c 'trap exit TERM; while ;; do certbot renew; sleep
12h & wait $$(!); done;'"
```

3. Posteriormente modificamos el archivo de configuración nginx.conf localizado dentro del directorio mongodb para mantener el funcionamiento del proxy y agregar lo del certificado como se muestra en el siguiente cuadro. Se debe de reemplazar dentro del siguiente archivo la etiqueta **<dominio>** por el nombre del dominio.

```
user  nginx;

events {
    worker_connections  1000;
}

http {
    upstream oauth2{
        server oauth20v2:9004;
    }
    upstream frontend{
        server frontend:3000;
    }
    upstream backend{
        server backend:3004;
    }

    server {
        listen 80;
        server_name <dominio>;

        location /.well-known/acme-challenge {
            root /var/www/certbot;
        }

        location /back {
            return 301 https://<dominio>/back/$request_uri;
        }

        location /secure {
            return 301 https://<dominio>/secure/$request_uri;
        }

        location / {
            return 301 https://<dominio>$request_uri;
        }
    }
}
```

```
server {

    listen 443 ssl;
    server_name <dominio>;
    ssl_certificate /etc/letsencrypt/live/<dominio>/fullchain.pem;
    ssl_certificate_key /etc/letsencrypt/live/<dominio>/privkey.pem;

    root /var/www/html;

    include /etc/letsencrypt/options-ssl-nginx.conf;
    ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem;

    location /.well-known/acme-challenge {
        root /var/www/certbot;
    }

    location /back {
        rewrite /back(/.*)$ $1 break;
        proxy_pass http://backend;
        proxy_http_version 1.1;
        proxy_set_header 'Access-Control-Allow-Origin' '*';
        proxy_set_header 'Access-Control-Allow-Methods' 'GET, POST,
OPTIONS';

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location /secure {
        rewrite /secure(/.*)$ $1 break;
        proxy_pass http://oauth2;
        proxy_http_version 1.1;
        proxy_set_header 'Access-Control-Allow-Origin' '*';
        proxy_set_header 'Access-Control-Allow-Methods' 'GET, POST,
OPTIONS';

        proxy_set_header 'Access-Control-Allow-Headers'
'DNT,User-Agent,X-Requested-With,If-Modified-Since,Cache-Control,Content-Type,Range
';

        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }

    location / {
```

```
        proxy_pass http://frontend$request_uri;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Por último, para hacer el proceso de generación del certificado SSL temporal y la solicitud hacia Let's Encrypt.

4. Crear un nuevo archivo nombrado `init-letsencrypt.sh` con el contenido del siguiente recuadro. Pero, reemplazando **<dominio>** y **<email>** por valores válidos.

Por ejemplo:

```
domains=(servicepdn.ga www.servicepdn.ga)
```

```
email="example@example.com"
```

```
#!/bin/bash

if ! [ -x "$(command -v docker-compose)" ]; then
    echo 'Error: docker-compose is not installed.' >&2
    exit 1
fi

domains=(<dominio>)
rsa_key_size=4096
data_path="./data/certbot"
email="<email>" # Adding a valid address is strongly recommended
staging=0 # Set to 1 if you're testing your setup to avoid hitting request limits

if [ -d "$data_path" ]; then
    read -p "Existing data found for $domains. Continue and replace existing certificate? (y/N) " decision
    if [ "$decision" != "Y" ] && [ "$decision" != "y" ]; then
```

```
        exit
    fi
fi

if [ ! -e "$data_path/conf/options-ssl-nginx.conf" ] || [ ! -e
"$data_path/conf/ssl-dhparams.pem" ]; then
    echo "### Downloading recommended TLS parameters ..."
    mkdir -p "$data_path/conf"
    curl -s
https://raw.githubusercontent.com/certbot/certbot/master/certbot-nginx/certbot_nginx/_internal/tls_configs/options-ssl-nginx.conf >
"$data_path/conf/options-ssl-nginx.conf"
    curl -s
https://raw.githubusercontent.com/certbot/certbot/master/certbot/certbot/ssl-dhparams.pem > "$data_path/conf/ssl-dhparams.pem"
    echo
fi

echo "### Creating dummy certificate for $domains ..."
path="/etc/letsencrypt/live/$domains"
mkdir -p "$data_path/conf/live/$domains"
docker-compose run --rm --entrypoint "\
    openssl req -x509 -nodes -newkey rsa:2048 -days 1\
        -keyout '$path/privkey.pem' \
        -out '$path/fullchain.pem' \
        -subj '/CN=localhost'" certbot
echo

echo "### Starting nginx ..."
docker-compose up --force-recreate -d nginx
echo

echo "### Deleting dummy certificate for $domains ..."
docker-compose run --rm --entrypoint "\
    rm -Rf /etc/letsencrypt/live/$domains && \
    rm -Rf /etc/letsencrypt/archive/$domains && \
    rm -Rf /etc/letsencrypt/renewal/$domains.conf" certbot
echo

echo "### Requesting Let's Encrypt certificate for $domains ..."
#Join $domains to -d args
domain_args=""
for domain in "${domains[@]"; do
    domain_args="$domain_args -d $domain"
done
```



```
# Select appropriate email arg
case "$email" in
  "") email_arg="--register-unsafely-without-email" ;;
  *) email_arg="--email $email" ;;
esac

# Enable staging mode if needed
if [ $staging != "0" ]; then staging_arg="--staging"; fi

docker-compose run --rm --entrypoint "\
  certbot certonly --webroot -w /var/www/certbot \
    $staging_arg \
    $email_arg \
    $domain_args \
    --rsa-key-size $rsa_key_size \
    --agree-tos \
    --force-renewal" certbot

echo

echo "### Reloading nginx ..."
docker-compose exec nginx nginx -s reload
```

5. Se necesita otorgar permisos de ejecución y después ejecutar.

```
$ chmod +x init-letsencrypt.sh
$ sudo ./init-letsencrypt.sh
```

Nota: Si no se recibe ningún mensaje de error, entonces el certificado SSL se ha generado correctamente.

6. Anexo 1 - Base de datos

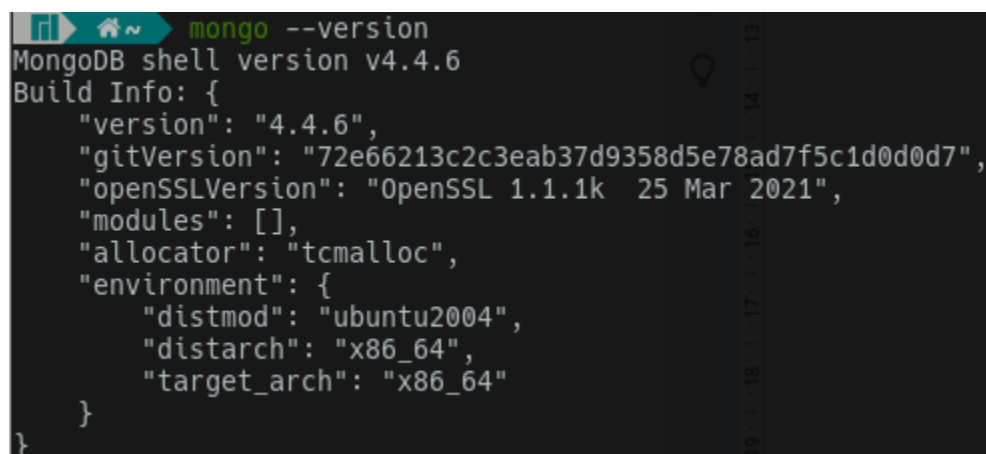
6.1. Consola

6.1.1. Conexión

Para conectarse a una base de datos MongoDB deberá tener instalado el cliente mongo, para validar que esté correctamente instalado ejecute el siguiente comando:

```
$ mongo --version
```

Deberá obtener una salida parecida a la siguiente imagen



```
mongo --version
MongoDB shell version v4.4.6
Build Info: {
  "version": "4.4.6",
  "gitVersion": "72e66213c2c3eab37d9358d5e78ad7f5c1d0d0d7",
  "opensslVersion": "OpenSSL 1.1.1k 25 Mar 2021",
  "modules": [],
  "allocator": "tcmalloc",
  "environment": {
    "distmod": "ubuntu2004",
    "distarch": "x86_64",
    "target_arch": "x86_64"
  }
}
```

Ahora que válido que está instalado el cliente de MongoDB, puede intentar realizar una conexión a una base de datos, para ello requiere tener la siguiente información mínima

Parámetro	Uso	Descripción
Host	--host	puede ser una IP o un dominio
Puerto	--port	por default es el 27017 , pero en algunos casos puede variar, si es el mismo puede omitir este parámetro en la conexión
Usuario	-u	Nombre de usuario
Contraseña	-p	Password del usuario
Base de autenticación	--authenticationDatabase	Es la base de datos donde fue registrado el usuario para poder acceder, regularmente es admin , en caso contrario es necesario especificar el parámetro

Con el conocimiento de los valores de los parámetros anteriores puede ejecutar el siguiente comando para realizar la conexión.

```
mongo --host <IP> --port 27017 --authenticationDatabase <admin> -u <usuario> -p
```

Si los datos ingresados son correctos tendrá una salida parecida a la siguiente:

```
MongoDB shell version v4.4.6
Enter password:
connecting to: mongodb://10.2.14.83:27017/?authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("dc1bea67-daa5-46b3-ae04-73f901827f83") }
MongoDB server version: 4.4.5
---
The server generated these startup warnings when booting:
  2021-04-23T00:18:58.700+00:00: /sys/kernel/mm/transparent_hugepage/enabled is 'always'. We suggest setting it to 'never'
  2021-04-23T00:18:58.701+00:00: /sys/kernel/mm/transparent_hugepage/defrag is 'always'. We suggest setting it to 'never'
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> |
```

Al final de la pantalla podrá observar un símbolo > que indica que puede introducir comandos a MongoDB.

6.1.2. Operaciones - Base de Datos

Para realizar cualquier operación dentro de MongoDB debe indicar sobre qué base de datos estará trabajando. Para saber las bases de datos disponibles ejecute el siguiente comando:

```
> show dbs
```

De las bases que despliegue el comando puede seleccionar una o en caso de que no exista puede crearla. Para cualquier caso el comando que debe ejecutar es el siguiente

```
> use <nombre de la base de datos>
```

Por ejemplo:

```
> use pruebas
```

Para saber en qué base de datos nos encontramos actualmente ejecutamos el siguiente comando

```
> db
```

NOTA: Una base de datos no será registrada dentro de mongo hasta que se cree la primera colección y dentro de esa colección se registre un documento. Por lo tanto, si sale si hacer nada en una nueva base de datos, cuando regrese esta no se encontrara dentro de la lista de base de datos del comando *show dbs*.

6.1.3. Operaciones - Colecciones

Para conocer las colecciones que tiene la base de datos en la que actualmente estamos trabajando debemos ejecutar el siguiente comando

```
> show collections
```

Le mostrará las colecciones actuales, en caso de no existir ninguna no le mostrará nada.

Para crear una colección se puede hacer de dos maneras, la primera con el comando **db.createCollection** y la segunda insertando un documento en la misma.

La diferencia es que la segunda opción garantiza que la colección existirá dentro de la base de datos, mientras que la primera solo la creará de forma temporal hasta que se realice algún registro en la misma.

Para crear una colección con el comando antes mencionado solo ejecute cómo en el siguiente ejemplo

```
> db.createCollection('coleccionPruebas');
```

Para eliminar una colección deberá ejecutar el siguiente comando

```
> db.coleccionPruebas.drop();
```

Tome nota que este procedimiento no se puede deshacer, y que al eliminar una colección se eliminan todos los documentos que existen en ella.

Para renombrar una colección porque no contiene el nombre necesario puede ejecuta el siguiente ejemplo

```
> db.coleccionPruebas.renameCollection('nuevoNombrePruebas')
```

6.1.4. Operaciones - Documentos

Para insertar un documento debemos ejecutar el siguiente comando de ejemplo

```
> db.nuevoNombrePruebas.insert({ "atributo": "este es un atributo",  
"atributo2": "este es otro atributo"});
```

Nota: si la colección no existe, cuando se inserta un documento, esta será creada de forma automática.

Para ver los documentos que contiene una colección debe ejecutar el siguiente comando de ejemplo:

```
> db.nuevoNombrePruebas.find();
```

Nota: Este comando le mostrará solo los primeros 20 registros, para continuar listado los demás deberá ejecutar el comando *it*.

Para eliminar un documento, debe especificar un valor de búsqueda para que se identifique en este caso usaremos el ObjectId de los que aparecen en la lista al hacer una búsqueda el comando quedaría cómo el siguiente ejemplo

```
> db.nuevoNombrePruebas.deleteOne({"_id" : ObjectId("60ad287d43f2f1844d9487fd")});
```

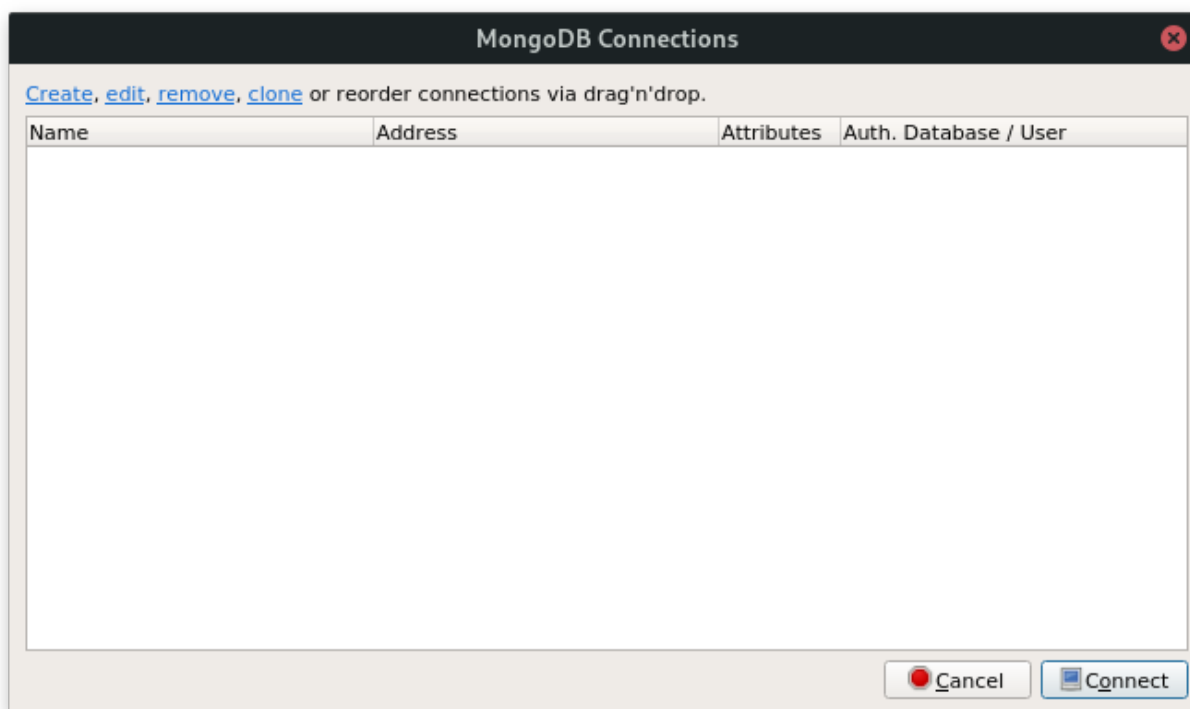
Para poder eliminar utilizando otros valores dentro del documento puede leer [la documentación de MongoDB](#).

6.2. GUI Robo3T

Robo3T es una aplicación gratuita de MongoDB, la puedes descargar del siguiente [link](#). Dentro de la página debe seleccionar la opción Download Robo 3T Only. Le pedirá que llene un formulario con datos de contacto, estos sirven para enviarle publicidad y actualizaciones sobre MongoDB, después de llenarlo podrá realizar la descarga de un zip o un exe en caso de Windows.

- Si escogió el EXE ejecútelo y siga los pasos de instalación con las configuraciones preestablecidas, al terminar la instalación deberá aceptar la licencia de usuario final y en la siguiente pantalla solo presionar el botón de Finish.
- Si escogió el ZIP, descomprímalo y ejecute el archivo robo3t.exe,
- Si escogió otro sistema operativo, al descomprimir encontrar una carpeta llamada bin, dentro de ella se encuentra el ejecutable robo3t que deberá abrir.

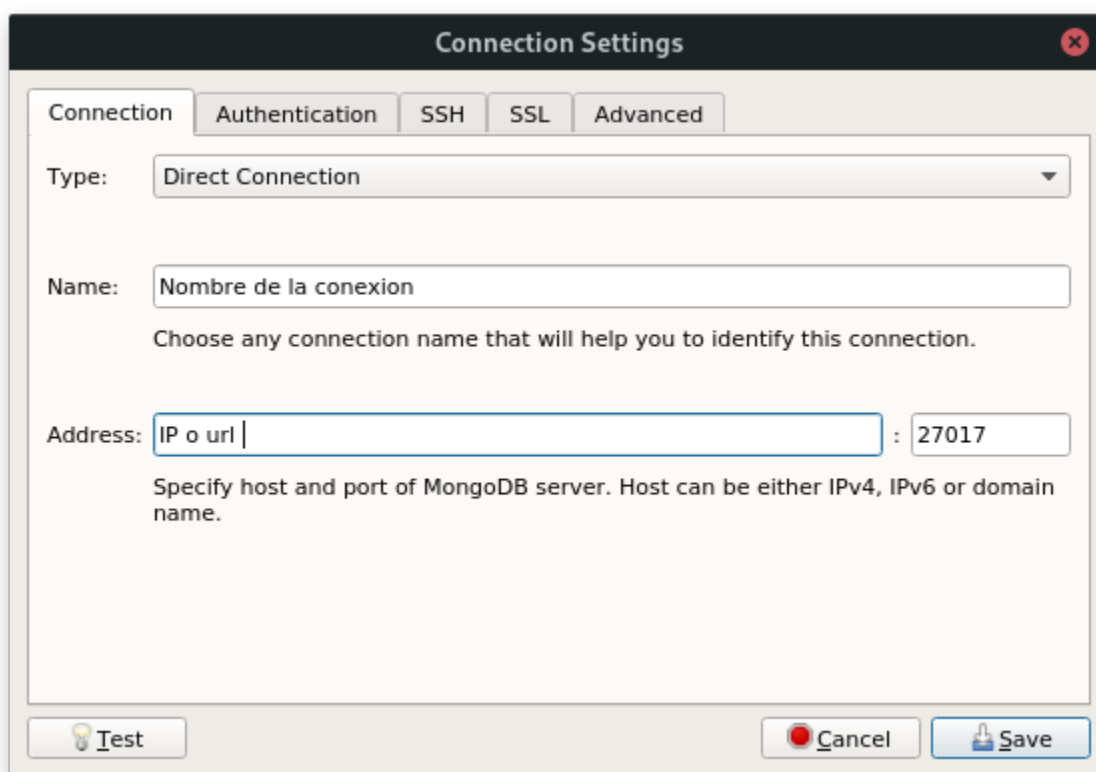
Al hacerlo le mostrará la siguiente pantalla.



Lo que indicara que está correctamente instalado.

6.2.1. Conexión

En la venta anterior tendrá que dar clic en el enlace azul que dice Create, para poder crear una nueva conexión a la base de datos. Le mostrará la siguiente ventana

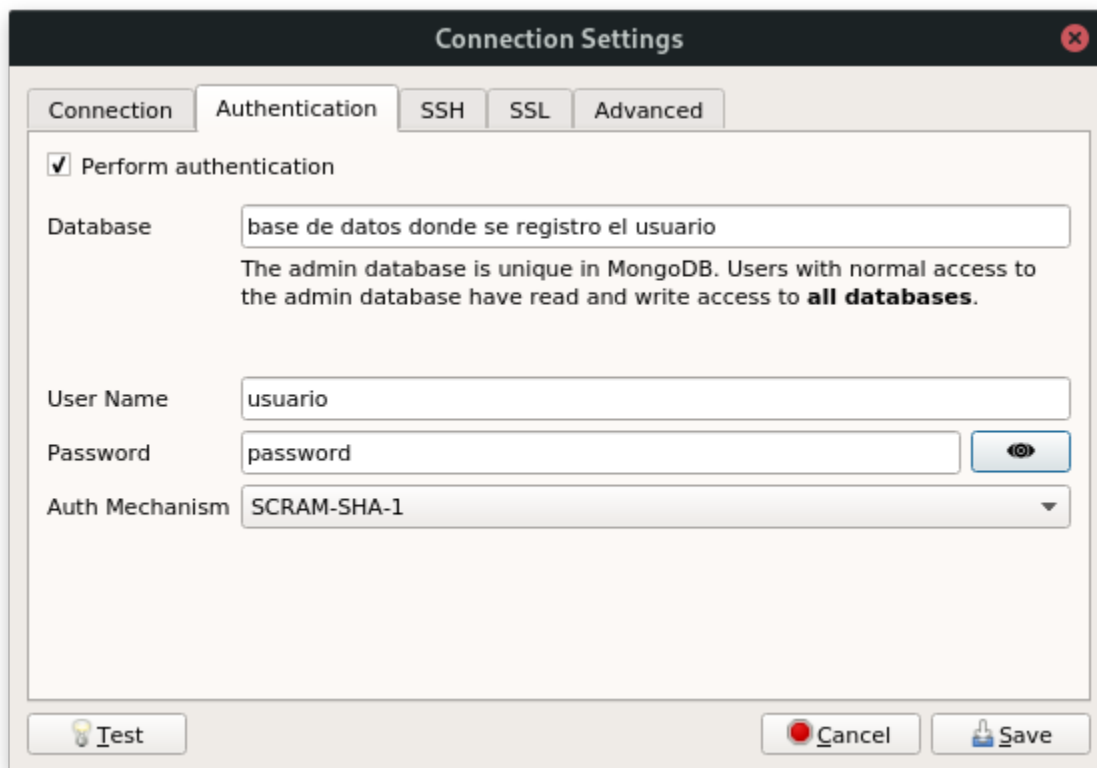


The screenshot shows a 'Connection Settings' dialog box with the following fields and options:

- Connection Settings** (Title bar)
- Tabs:** Connection (selected), Authentication, SSH, SSL, Advanced
- Type:** Direct Connection (dropdown menu)
- Name:** Nombre de la conexion (text input field)
- Address:** IP o url (text input field) : 27017 (port input field)
- Instructions:** Choose any connection name that will help you to identify this connection. Specify host and port of MongoDB server. Host can be either IPv4, IPv6 or domain name.
- Buttons:** Test (highlighted), Cancel, Save

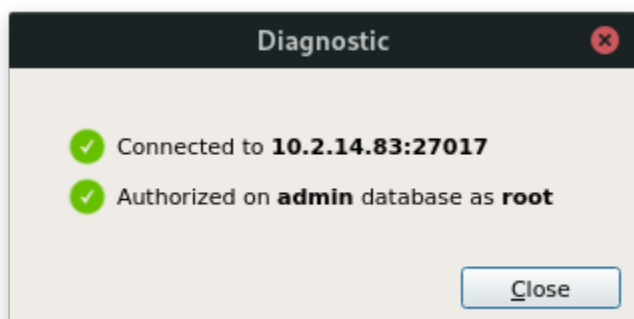
En la primera pestaña deberá colocar el nombre con el que se va a identificar la nueva conexión, además de la IP/dominio del servidor y el puerto de ser necesario.

En la pestaña de Authentication, debe colocar el nombre de la base de datos en la que se registró el usuario, el usuario y su password, el valor de Auth Mechanism se queda con el valor default



The screenshot shows the 'Connection Settings' dialog box with the 'Authentication' tab selected. The 'Perform authentication' checkbox is checked. The 'Database' field contains 'base de datos donde se registro el usuario'. Below this field, a warning message states: 'The admin database is unique in MongoDB. Users with normal access to the admin database have read and write access to **all databases**.' The 'User Name' field contains 'usuario', the 'Password' field contains 'password', and the 'Auth Mechanism' dropdown is set to 'SCRAM-SHA-1'. At the bottom, there are buttons for 'Test', 'Cancel', and 'Save'.

Al terminar, puede ejecutar el botón Test que se encuentra el parte inferior izquierda para probar que todos los datos son correcto y que es posible conectarse a MongoDB.

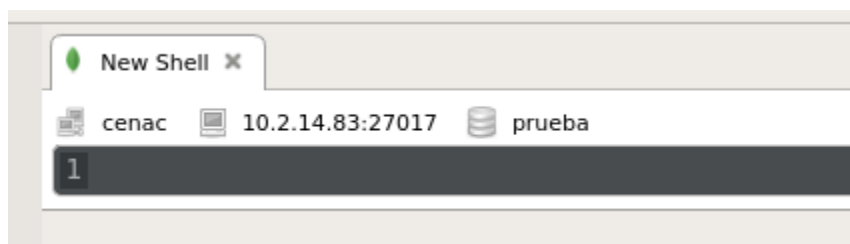


The screenshot shows the 'Diagnostic' dialog box with two green checkmarks indicating a successful connection and authorization. The first message is 'Connected to 10.2.14.83:27017' and the second is 'Authorized on admin database as root'. A 'Close' button is located at the bottom right.

Deberá tener un resultado similar al anterior.

6.2.2. Operaciones

Para ejecutar las mismas operaciones que se realizaron en la parte anterior únicamente necesita es dar clic derecho sobre la base de datos a utilizar y seleccionar la opción Open Shell.



En el Shell que se abre puede ejecutar los mismos comandos de la sección anterior, también puede hacer uso de los menús de clic derecho sobre los elementos de la base de datos

