

**Laboratorio de sistemas
operativos y redes**

Traccar

**Érica Gerez,
Juan Carlos Francisco Stabile.**

Contenido

1. ¿Qué es Traccar?
2. Introducción
3. Elementos necesarios
4. Instalación
5. Referencias

1. ¿Qué es Traccar?

Traccar es una plataforma de monitoreo de posiciones globales que una vez instalado en una máquina, convierte a esta misma en un servidor de localización, mediante el cual es posible monitorear el seguimiento de una ubicación GPS de un cliente específico.

2. Introducción

El objetivo de nuestro trabajo es demostrar el funcionamiento de la plataforma de rastreo por GPS *Traccar*. Para esto mismo, se procederá a instalar un servidor *Traccar* sobre un máquina que tenga un sistema operativo GNU-Linux. Para ello se realizarán las configuraciones pertinentes que se detallarán en la sección *Instalación* de este mismo informe. Finalmente haremos uso de la app cliente *Traccar* en un celular con sistema operativo Android. Esta última mencionado se hará con el fin de poner a prueba las funcionalidades de seguimiento proveída a través de la interacción entre el servidor y su cliente, logrando así el propósito de este trabajo.

Traccar está escrito en Java puro, por lo que se puede ejecutar en cualquier plataforma que admita Java SE. Si no se proporciona un instalador especial para su plataforma o algo no funciona, siempre puede instalar y ejecutar el servidor de rastreo GPS Traccar manualmente.

3. Elementos necesarios

→ Celular (u otro dispositivo móvil) que cuente con:

- ◆ Sistema operativo Android
- ◆ Localización GPS.
- ◆ Acceso a internet.

→ Computadora en domicilio que cuente con:

- ◆ Sistema operativo GNU-Linux que soporte Java SE.
- ◆ Acceso a internet.
- ◆ Acceso de configuración al router domiciliario.

4. Instalación

La instalación la llevaremos a cabo en un servidor debian 9.11, cpu AMD 5150 y memoria 3.3G.

```
$ lsb_release -a
Description:      Debian GNU/Linux 9.11 (stretch)

$ cat /proc/cpuinfo | sort -r | uniq | grep -e 'model name' -e 'cpu
cores' -e 'address'
model name       : AMD Athlon(tm) 5150 APU with Radeon(tm) R3
cpu cores        : 4
address sizes     : 40 bits physical, 48 bits virtual

$ cat /proc/meminfo | grep MemTotal
MemTotal:        3479660 kB
```

4.1 Configuración de cron

Con el fin de que el cliente pueda acceder a un servidor que esté instalado en un domicilio, usaremos el servicio de *ducker DNS*, el cual es un proveedor de DNS dinámico. Para esto, seguiremos la siguiente instalación en una máquina

En principio verificamos que la máquina tenga instalado *curl* y *cron* :

```
$ apt policy cron
curl:
  Instalados: 7.52.1-5+deb9u9
  Candidato:  7.52.1-5+deb9u9
  Tabla de versión:
*** 7.52.1-5+deb9u9 500
    500 http://deb.debian.org/debian stretch/main amd64 Packages
    500 http://security.debian.org/debian-security
stretch/updates/main amd64 Packages
    100 /var/lib/dpkg/status
cron:
  Instalados: 3.0p11-128+deb9u1
  Candidato:  3.0p11-128+deb9u1
  Tabla de versión:
*** 3.0p11-128+deb9u1 500
```

```
500 http://deb.debian.org/debian stretch/main amd64 Packages
100 /var/lib/dpkg/status
```

De no estar instalado alguno de ellos, podemos corregir la situación fácilmente con:

```
# apt install curl
```

y/o

```
# apt install cron
```

Verificamos que esté corriendo *cron* en la máquina con:

```
$ systemctl status cron
● cron.service - Regular background program processing daemon
   Loaded: loaded (/lib/systemd/system/cron.service; enabled; vendor
  preset: enabled)
   Active: active (running) since Sun 2019-12-01 23:27:51 -03; 1 day
 10h ago
     Docs: man:cron(8)
    Main PID: 307 (cron)
      Tasks: 1 (limit: 4915)
    CGroup: /system.slice/cron.service
           └─307 /usr/sbin/cron -f
```

o con:

```
$ ps -fe | grep cr[o]n
root      305      1  0 nov16 ?          00:00:01 /usr/sbin/cron -f
```

En caso que no esté corriendo podemos iniciarlo con:

```
# systemctl start cron
```

Con *cron* corriendo seguimos adelante con la instalación del script, creamos entonces un directorio en */opt* donde residirá el script que con la asistencia de *cron* mantendrá la IP actualizada del servidor DNS, detectando nuestra IP pública.

```
# mkdir /opt/duckdns ; cd /opt/duckdns
```

Acabamos de crear el directorio que contendrá el script, ahora debemos agregar el código en el archivo del script.

```
# echo `echo url =  
  "https://www.duckdns.org/update?domains=tplabo&token=5cd9011d-2eb  
  2-4370-a857-4a9ff121607c&ip=" | curl -k -o  
  /opt/duckdns/tplabo.log -K -' > /opt/duckdns/tplabo.sh
```

Le asignamos a nuestro script permisos de ejecución.

```
# chmod 700 tplabo.sh
```

Podríamos probar el script ejecutándolo con:

```
# ./tplabo.sh
```

Si tras la respuesta a esta ejecución (salida a consola de *cron*) el archivo `tplabo.log` contiene un `OK` todo funcionó correctamente. En caso que el contenido del log sea `KO` significa que el script se pudo comunicar con nuestro proveedor gratuito de DNS y algo salió mal. Probablemente sea pertinente verificar *dominio* y *token* en el script.

En cualquier otro caso corregiremos conforme al error que obtengamos.

Una vez que sabemos que nuestro script ejecuta y actualiza la IP remota en nuestro proveedor, a continuación actualizaremos el proceso *cron* haciendo que corra `/opt/duckdns/tplabo.sh` cada cinco minutos. (nota: si en tu instalación aun no esta configurado el editor por defecto, en esta ocasión se te pedirá que lo determines)

```
# crontab -e
```

Se abrirá un editor. Agregaremos aquí la siguiente entrada:

```
*/5 * * * * /opt/duckdns/tplabo.sh > /dev/null 2>&1
```

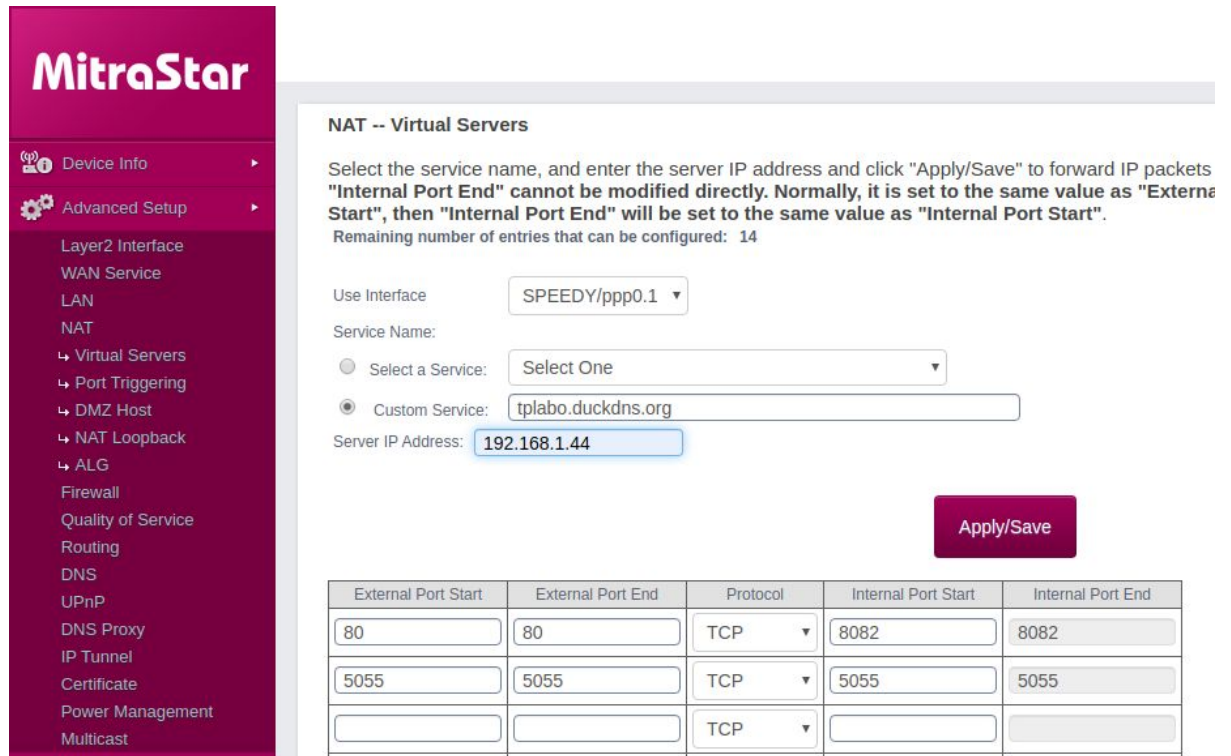
Salimos del editor, escribiendo nuestro agregado. De aquí en más cada cinco minutos nuestra la IP pública de nuestra instalación se asociará con el nombre de dominio `tplabo.duckdns.org`.

4.2 Seteo del NAT en el router

En el router tenemos que configurar dos ports. Uno para acceder a la interface web del servidor y otro para que los dispositivos que se reporten al servidor transmitan sus datos de seguimiento desde la nube. La imagen a continuación describe la configuración otorgada al router. El server donde instalaremos el *traccar server* tiene la IP 192.168.1.44. Se establece:

- el port 80 sobre la IP pública hacia el port 8082 sobre la IP interna 192.168.1.44

- el port 5055 sobre la IP pública hacia el mismo port sobre la IP interna 192.168.1.44



MitraStar

Device Info

Advanced Setup

Layer2 Interface

WAN Service

LAN

NAT

Virtual Servers

Port Triggering

DMZ Host

NAT Loopback

ALG

Firewall

Quality of Service

Routing

DNS

UPnP

DNS Proxy

IP Tunnel

Certificate

Power Management

Multicast

NAT -- Virtual Servers

Select the service name, and enter the server IP address and click "Apply/Save" to forward IP packets
"Internal Port End" cannot be modified directly. Normally, it is set to the same value as "External Port Start", then "Internal Port End" will be set to the same value as "Internal Port Start".
 Remaining number of entries that can be configured: 14

Use Interface: SPEEDY/ppp0.1

Service Name:

Select a Service: Select One

Custom Service: tplabo.duckdns.org

Server IP Address: 192.168.1.44

Apply/Save

External Port Start	External Port End	Protocol	Internal Port Start	Internal Port End
80	80	TCP	8082	8082
5055	5055	TCP	5055	5055
		TCP		

4.3 Instalación server traccar

Una vez hecho esto haremos la instalación del servidor Traccar sobre un Debian. Para ello seguiremos los siguientes pasos:

Ir a <https://www.traccar.org/download/> y descargar el archivo zip correspondiente a la arquitectura. O desde la línea de comandos (con por ejemplo):

```
$ wget
https://github.com/traccar/traccar/releases/download/v4.6/traccar-linux-64-4.6.zip
```

Descomprimir lo descargado y, como root, correr traccar.run, desde el directorio donde se extrajo el archivo comprimido:

```
$ unzip traccar-linux-64-4.6.zip
$ su
# ./traccar.run
Creating directory out
Verifying archive integrity... 100% All good.
Uncompressing traccar 100%
```

```
Created symlink
/etc/systemd/system/multi-user.target.wants/traccar.service →
/etc/systemd/system/traccar.service.
```

Este script acaba de instalar el systemd unit file que permitirá que el servicio de traccar se levante desde el arranque. O que ahora nosotros lo levantemos manualmente y/o verifiquemos el estado del servicio. Entonces vamos a levantar el servicio con:

```
# systemctl start traccar
# systemctl status traccar
● traccar.service - traccar
   Loaded: loaded (/etc/systemd/system/traccar.service; enabled; vendor
  preset: enabled)
   Active: active (running) since Tue 2019-12-03 11:02:03 -03; 1s ago
 Main PID: 49434 (java)
    Tasks: 23 (limit: 6743)
   Memory: 138.3M
    CGroup: /system.slice/traccar.service
            └─49434 /opt/traccar/jre/bin/java -jar tracker-server.jar
  conf/traccar.xml
```

Correrlo como un usuario no-root (desde <https://www.traccar.org/linux/>)

Correr servicios como *root* no es una buena práctica.

Para que Traccar sea fácilmente instalable, decidimos no atar Traccar a correr con un usuario específico. De todas formas recomendamos encarecidamente no correr Traccar como *root*.

Usted puede lograr esto agregando un Drop-In en systemd al traccar.service. Siga los siguientes pasos:

```
# mkdir /etc/systemd/system/traccar.service.d/
# touch /etc/systemd/system/traccar.service.d/run-as-user.conf
```

El contenido de *run-as-user.conf* configura el usuario que usará traccar como el deseado por usted. Tenga en mente que el directorio donde reside traccar debe ser leíble y escribible por este usuario.

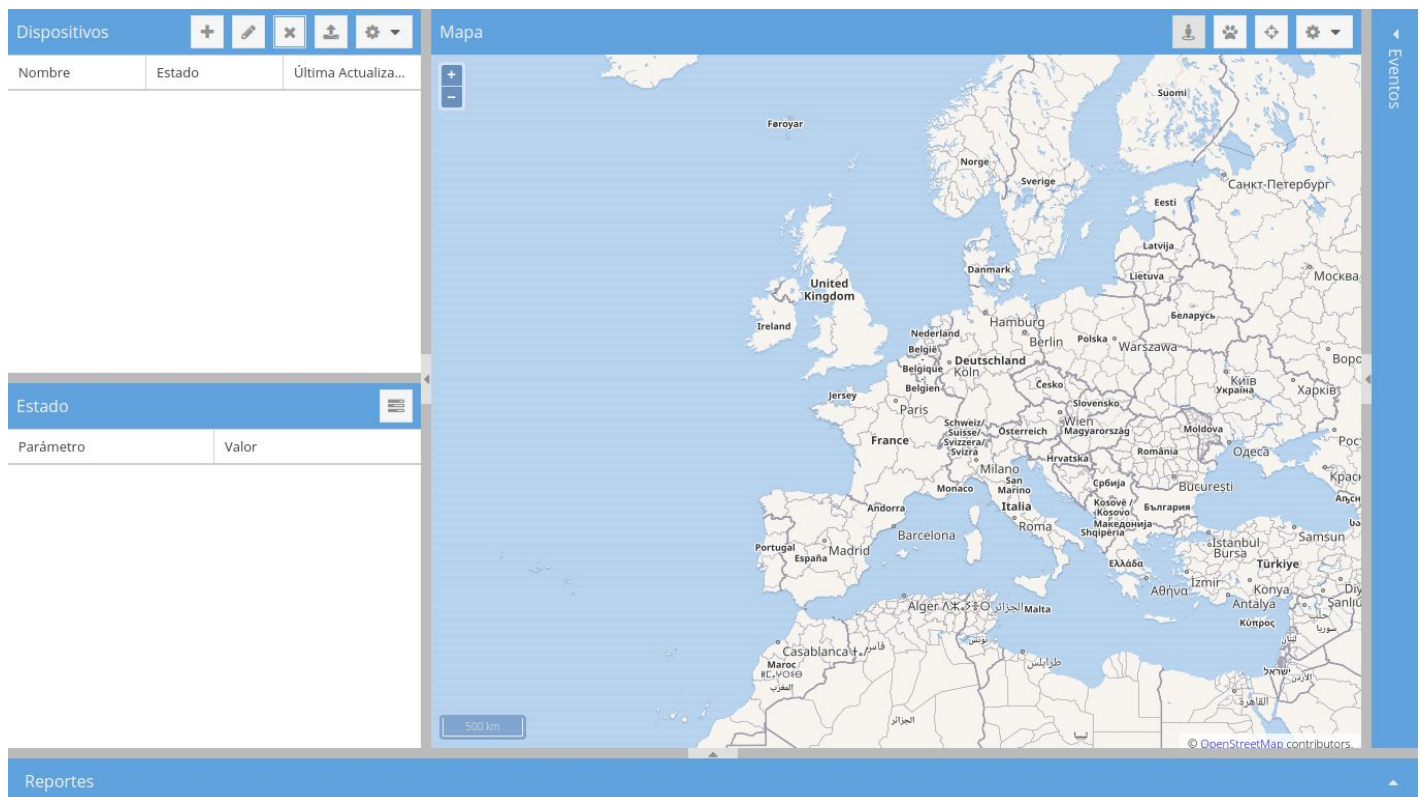
```
[Service]
User=traccar
Group=traccar
```


Ahora que el servicio está levantado podemos abrir el navegador e ir hasta <http://localhost:8082/>, ya que allí por defecto se levanta la interfaz web de Traccar. Crear e ingresar con un usuario y contraseña

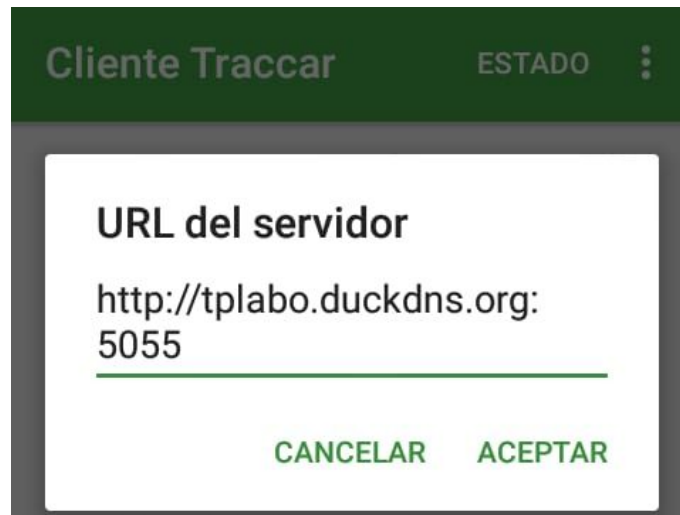


The image shows the Traccar web interface login form. It features the Traccar logo at the top left. Below the logo, there are four input fields: 'Idioma:' with a dropdown menu set to 'Español', 'Email:' with the text 'coquitas@mail.here', 'Contraseña:' with masked characters, and 'Recordar:' with an unchecked checkbox. At the bottom right, there are two buttons: 'Registrar' and 'Ingresar'.

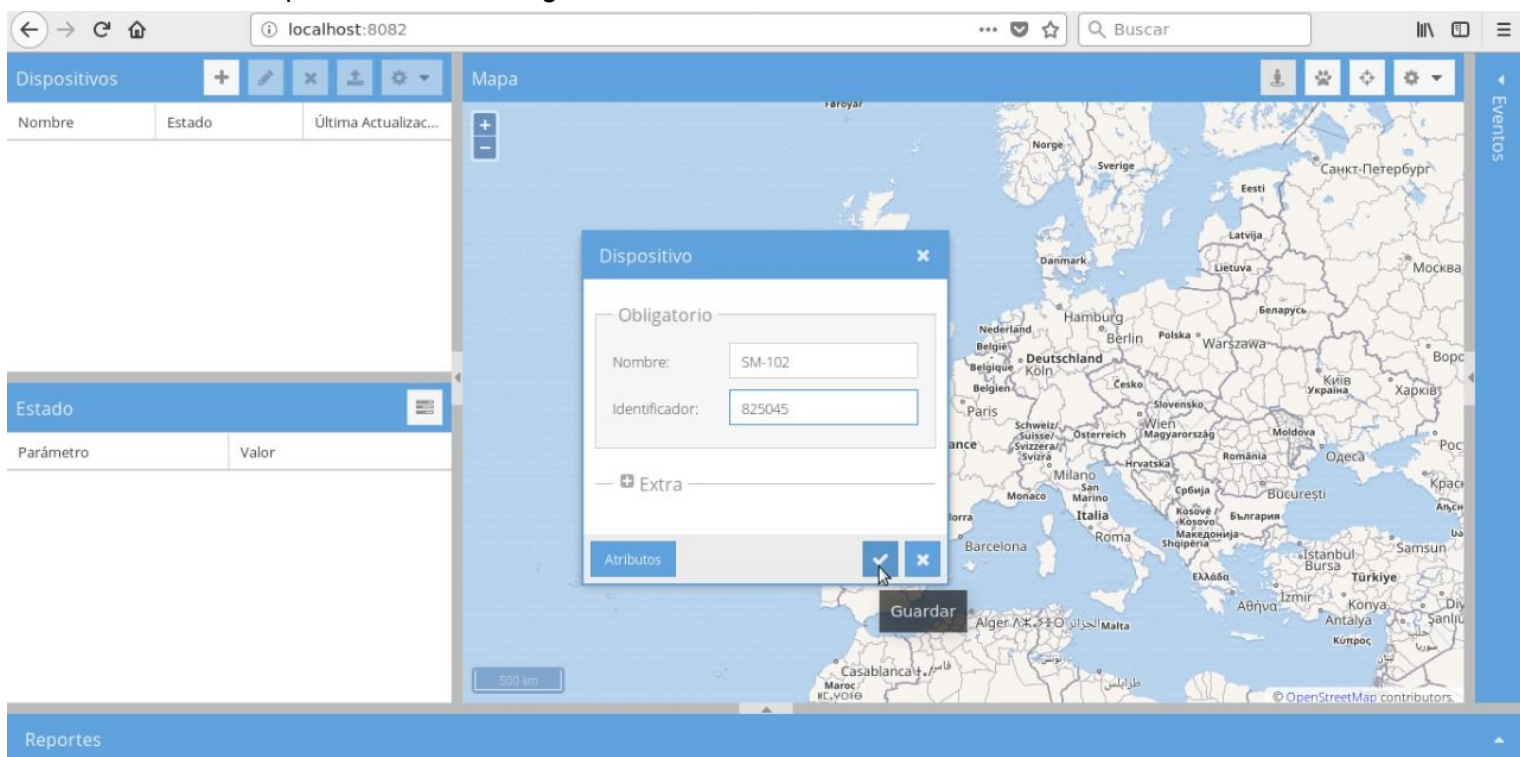
Se levantará la siguiente vista:



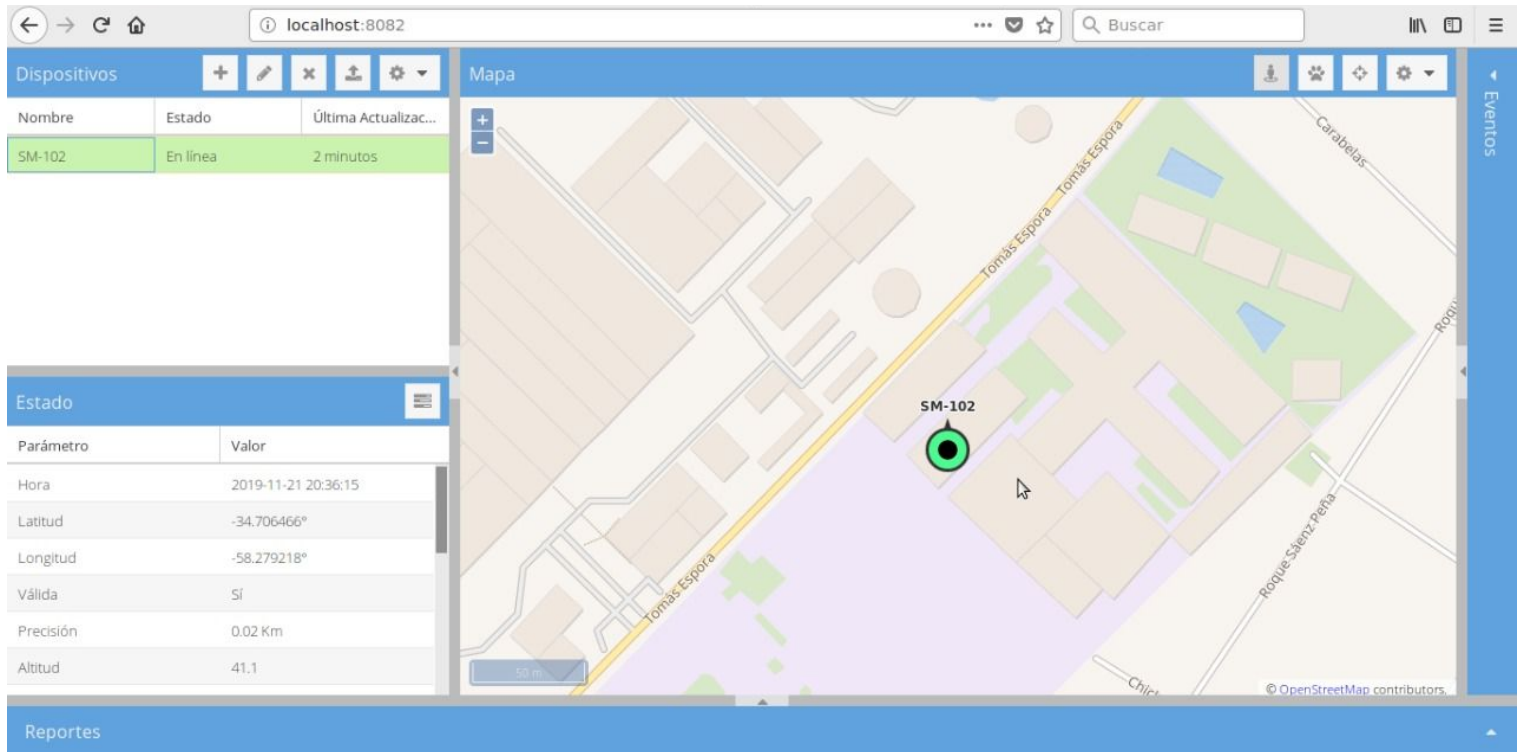
En el celular a utilizar, se deberá descargar la aplicación Traccar disponible en Play Store y configurar el campo *URL del servidor* con el dominio provisto por ducker



Regresando a <http://localhost:8082/>, agregaremos un nuevo dispositivo como se muestra en la siguiente imagen. El campo identificador, es provisto por la aplicación móvil mientras que el nombre lo elegimos nosotros mismos



Si la conexión fue exitosa, Traccar ya estará detectando la ubicación del dispositivo móvil



4.4 Instalación de traccar desde los fuentes

Clonar el repo desde github

```
$ git clone --recursive https://github.com/traccar/traccar.git
```

Ejecutar desde el directorio traccar

```
$ cd traccar ; ./gradlew
```

Que convierte el actual proyecto de *maven* a *gradle*. Para luego sí construir la aplicación.

```
$ gradle build
```

Ahora solo queda copiar los archivos sobre una instalación funcional o crear la instalación. En este segundo caso se presenta el problema de no contar con la herramienta con la que se genera el archivo. Vamos a omitir la creación de este archivo copiando este, desde la instalación resultante con el método “.zip”.

La secuencia de comandos para copiar los archivos resultantes de la construcción es como sigue.

Debido a que en la instalación desde los fuentes no contamos con el *jre* debemos referenciar a la máquina virtual de una instalación de java existente.

```
$ export SED_JAVA='s/\\opt\\traccar\\jre\\bin\\java\\/usr\\bin\\java/g'
$ sed -i $SED_JAVA setup/traccar.service
```

Aquí es cuando omitiremos la creación del archivo *app.min.js* agregando una copia desde una instalación anterior.

```
$ cp ../app.min.js traccar-web/web
```

Creamos directorios propios de la instalación y no creados en la construcción, *conf* y *logs*

```
# mkdir -p /opt/traccar/conf
# mkdir -p /opt/traccar/logs
```

Hacemos la sucesiva copia de los archivos desde el directorio de creación a los destinos en la instalación.

```
# cp -r target/* /opt/traccar
# cp setup/traccar.service /opt/traccar
# cp -r traccar-web/web /opt/traccar
# cp setup/*.xml /opt/traccar/conf
# cp -r schema /opt/traccar
# cp -r templates /opt/traccar
```

Cambiamos para *grupo* y *otros* los permisos a lectura (r) y ejecución/búsqueda solo si el archivo es un directorio o ya tiene permiso de ejecución para algún usuario (X), en el directorio de instalación:

```
# chmod -R go+rX /opt/traccar
```

Ahora instalamos el *unit file systemd*.

```
# mv /opt/traccar/traccar.service /etc/systemd/system
# chmod 664 /etc/systemd/system/traccar.service
```

Y finalmente habilitamos el servicio con *systemd*.

```
# systemctl daemon-reload
# systemctl enable traccar.service
```

4.5 Problemas encontrados

Al instalar desde los fuentes la construcción de la interface web depende de una herramienta no libre, *sencha*. Este comando es necesario para generar un archivo `.js: app.min.js` . De todas formas se puede simular la construcción copiando este archivo desde una instalación zip.

<https://github.com/traccar/traccar-web/issues/592>

Adicionalmente si no tenemos instalado openJDK 1.8 podemos hacerlo con:

```
# apt install openjdk-8-jre
```

5. Referencias:

<http://www.duckdns.org/>

<https://www.traccar.org/>

<https://www.traccar.org/protocols/>