

4X METRO

CENTRO DE INNOVACIÓN INDUSTRIAL

Dr. Raúl A. Trejo Ramírez

rtrejo@megahabilidades.mx

[@hipopotamo](#)

cii4xmetro.com

4 – PROCESO DE INFORMACIÓN

PRÁCTICA: PLATAFORMA CARRIOTS

INTRODUCCIÓN A CARRIOTS

1. ¿Qué es Carriots?
2. Construyendo un proyecto de Internet de las Cosas.
3. Dispositivos.
4. Plataforma.
5. Frontend.
6. Integración.
7. Beneficios principales de Carriots.
8. ¿Qué sigue después?



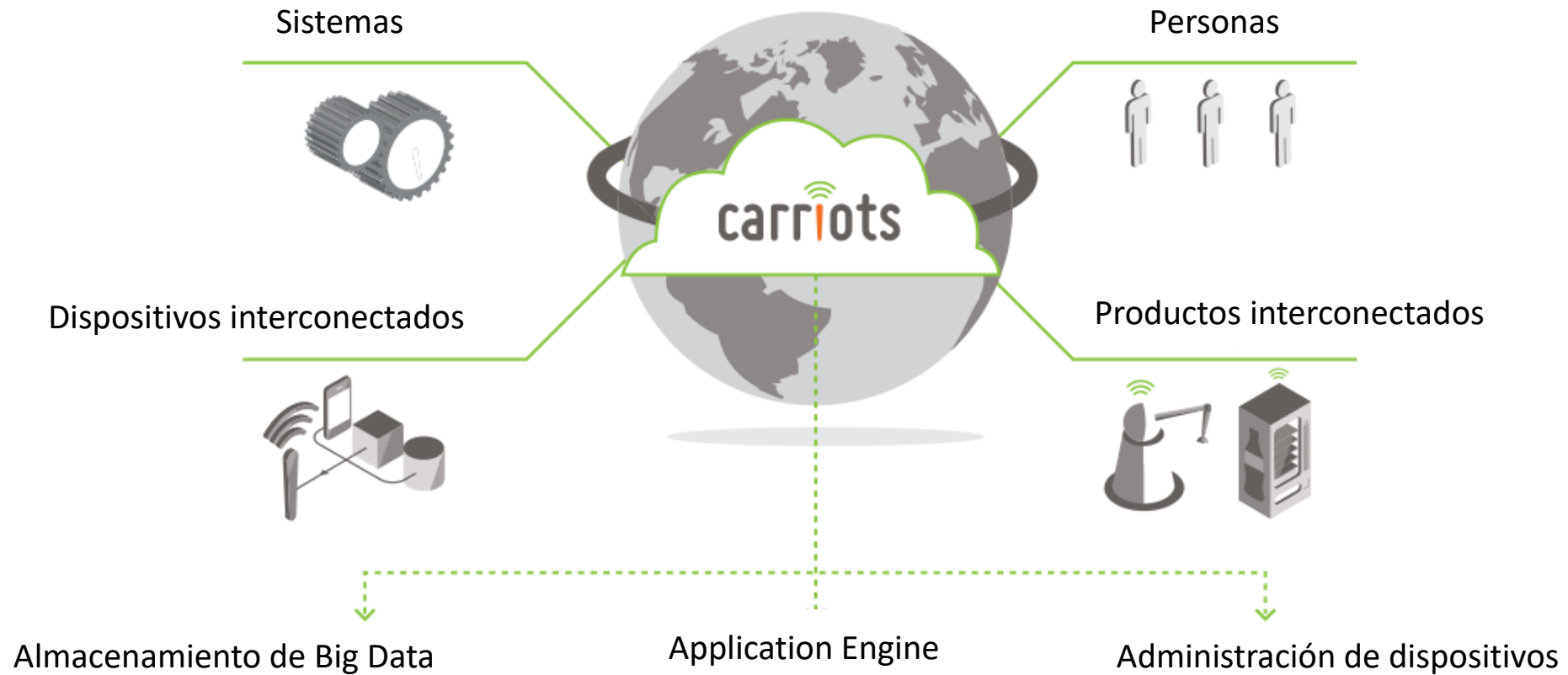
¿QUÉ ES CARRIOTS?

- Carriots es una plataforma de servicio (Paas)
- Diseñada para el Internet de las Cosas (IoT) y Proyectos máquina a máquina (M2M)

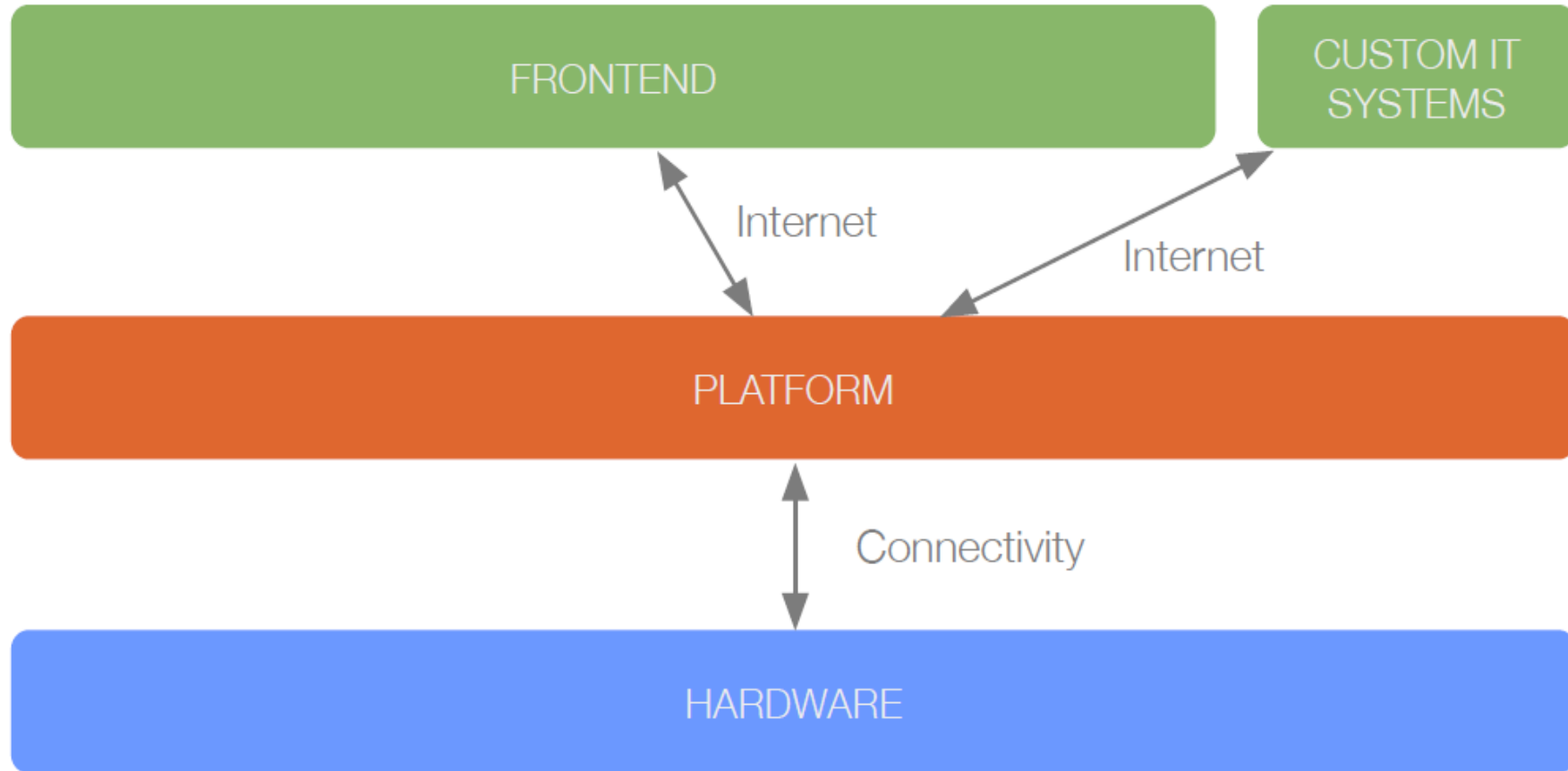
LO QUE PUEDO HACER CON CARRIOTS

- **Recolectar y almacenar** información proveniente de cualquier dispositivo
- **Construir soluciones** robustas con el Carriots Application Engine
- **Desplegar y escalar** desde prototipos muy pequeños a miles de dispositivos

LO QUE PUEDO HACER CON CARRIOTS



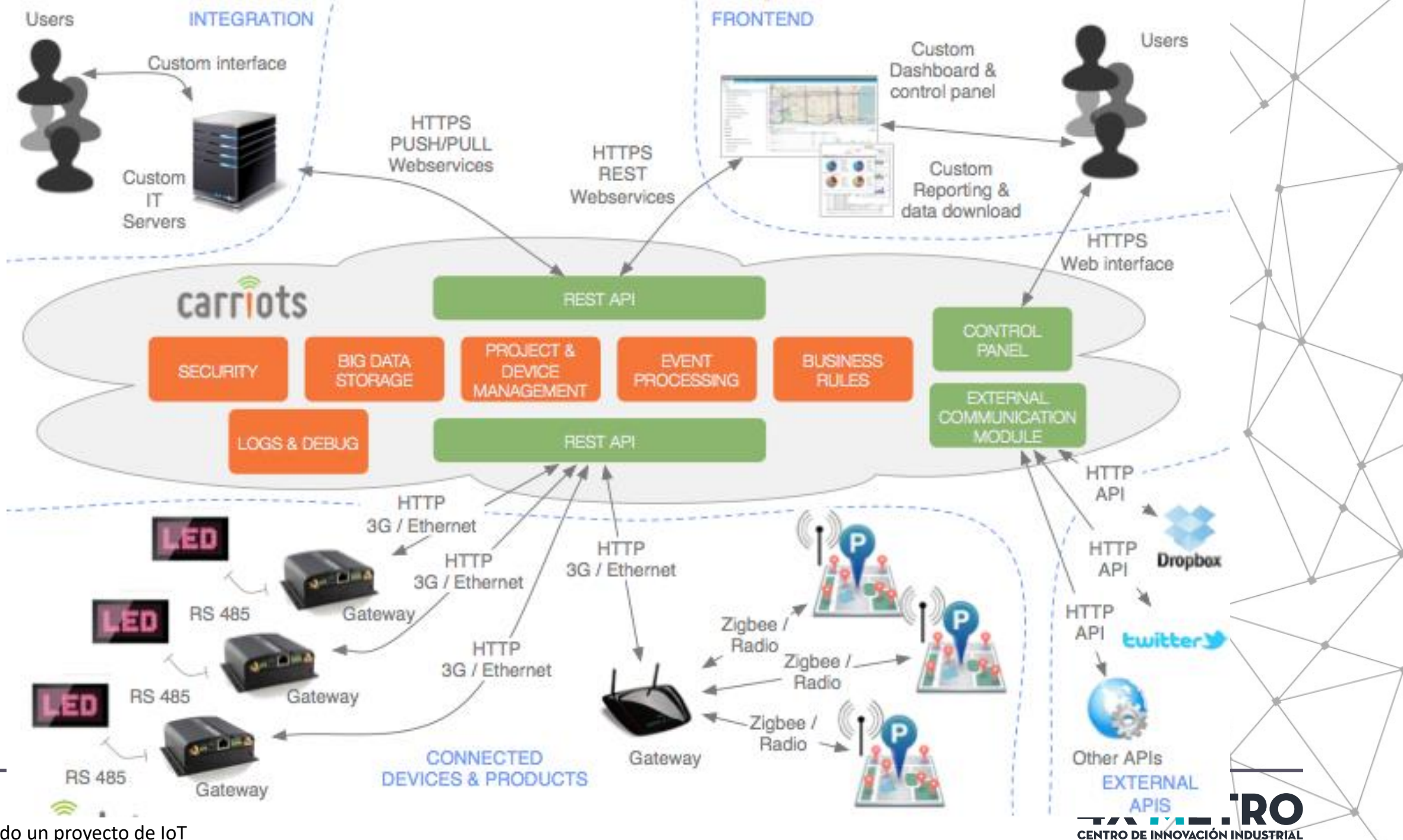
CONSTRUYENDO UN PROYECTO DE IOT



PRESENTACIÓN
(Monitoreo y Reporte)

APLICACIÓN
(Bases de datos y relaciones)

SENSORES Y ACTUADORES
(Cosas)



Construyendo un proyecto de IoT

CONSTRUYENDO UN PROYECTO DE IOT

¿Complejo o poderoso?

Veamos los elementos de un proyecto de IoT siguiendo un ejemplo:

Caso de un **Estacionamiento Inteligente**

DISPOSITIVOS

- Tienen **sensores** de recopilación de datos, ej: la detección magnética de un coche en un estacionamiento.
- Tienen **actuadores** para controlar cosas, ej: pantallas de LEDs en las calles.
- Necesitan **conectividad** a internet, ej: una puerta de enlace o módem 3G integrado.

PLATAFORMA

- Almacenar todos los **datos**: BD de big data donde se pueden realizar consultas y análisis de datos.
- Contiene y ejecuta toda la **lógica de la aplicación**: reglas, alarmas, etc.
- Proporciona **dispositivo y su software de gestión**, ej: aprovisionamiento de _dispositivos, habilitar y deshabilitar dispositivos, cambiar el firmware, etc.

FRONTEND

- Para que una aplicación se pueda usar el frontend proporciona la **interfaz para los usuarios**, por ejemplo: panel de control personalizado, tablero de instrumentos, informes, etc.

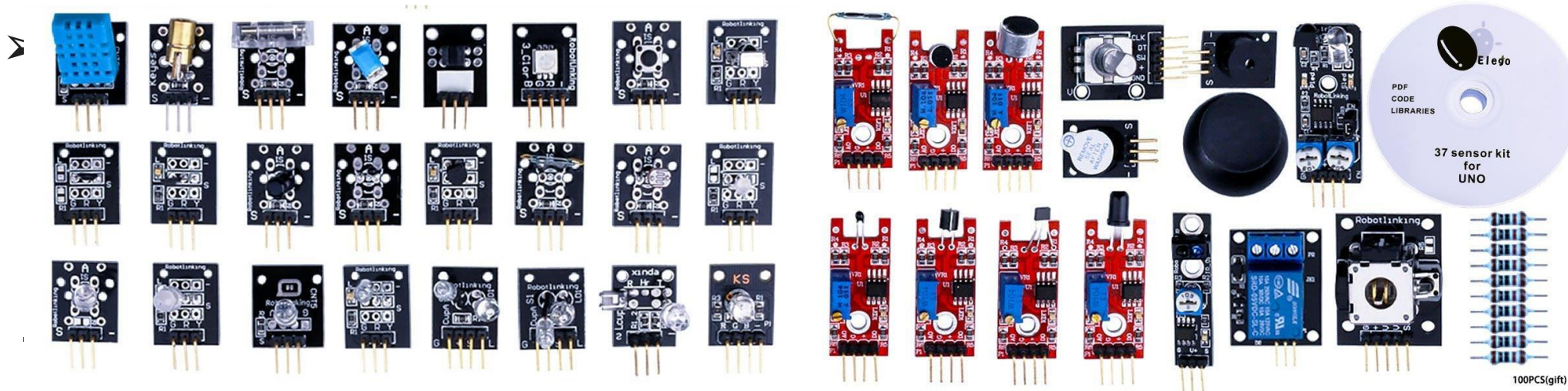
INTEGRACIÓN

- Carriers se puede integrar con otros sistemas y **empujar (PUSH) o jalar (PULL) datos** a/desde CRMs, ERPs, o cualquier API HTTP disponible, ej: el COI de IBM, Dropbox, Zoho, Twitter, etc.

DISPOSITIVOS

SENSORES Y ACTUADORES

- Los sensores recogen los datos que deben ser leídos por un dispositivo. Pueden medir la temperatura, la presencia o no de algo, el viento, campos magnéticos, energía, flujo de líquidos, la calidad del aire, vibraciones, posición geográfica, altitud, presión, etc.



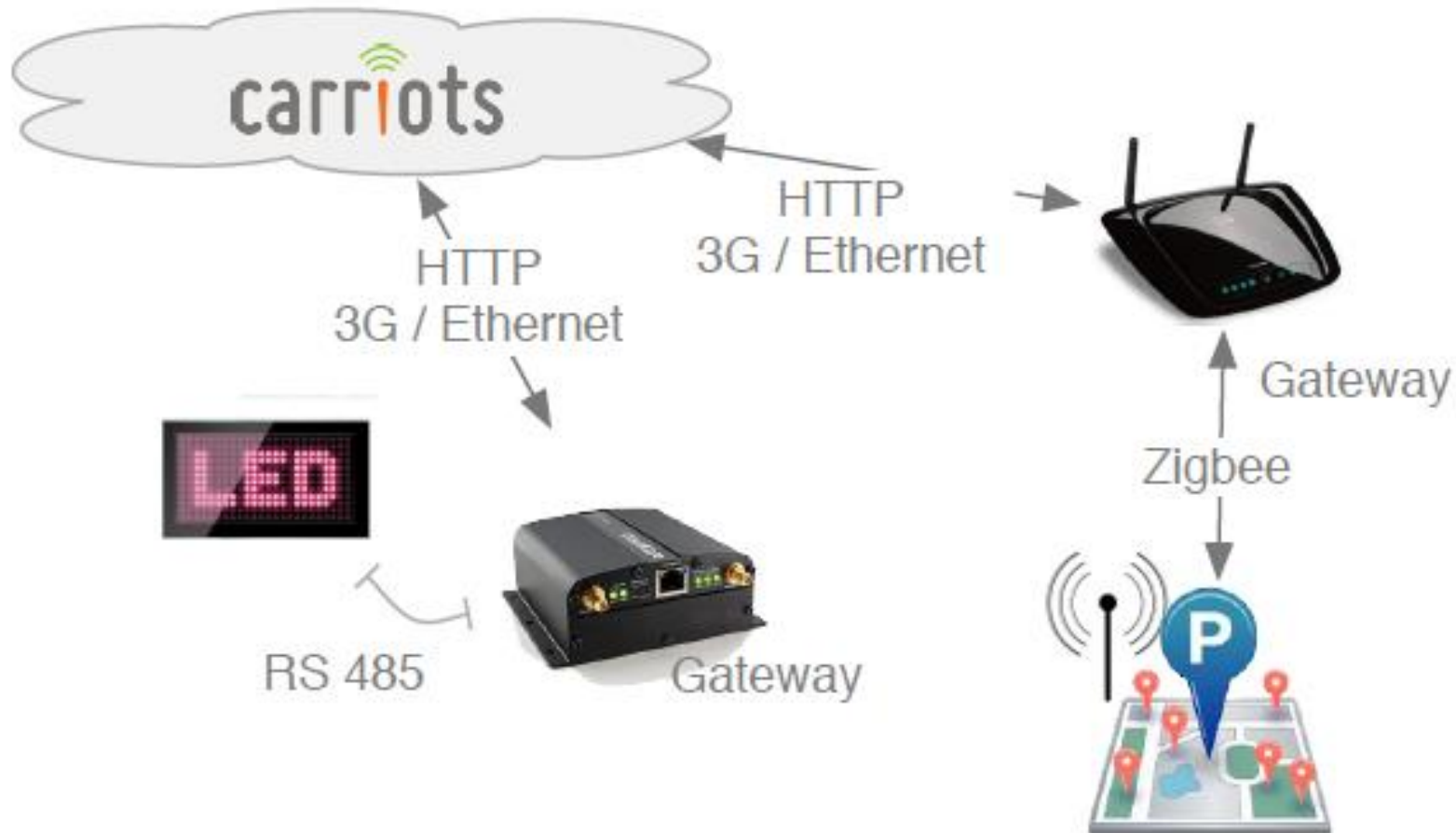
DISPOSITIVOS

CONECTIVIDAD

- Para que un proyecto pueda clasificarse como uno de IoT, es necesario que los dispositivos se puedan conectar al internet para interactuar.
- Conectividad Independiente
 - 3G / GPRS modems incluidos en los dispositivos
- Conectividad de puertos + Dispositivos
 - Serial (ej. RS232, RS485) o radio (ej. 868MHz) comunicación dispositivo-puerto.
- Red de sensores + puertos
 - Comunicación WiFi o ethernet local del dispositivo-puerto.

PROTOCOLO DE COMUNICACIÓN

- Los dispositivos interactúan con Carriots mediante el protocolo estándar HTTP / HTTPS para la capa del internet. Para la capa de apps (REST API) utilizan JSON



EJEMPLO: mandar datos del lugar del estacionamiento desde un dispositivo.

HTTP request

POST /streams HTTP/1.1

Host: api.carriots.com

Accept: application/json

User-Agent: place_524@smartparking

Content-Type: application/json

carriots.apikey:98346673a637...5a0d83045425407ab4

Content-Length: 182

Connection: close

REST API URL (POST request)

JSON FORMAT

SECURITY (APIKEY)

Data (PAYLOAD)

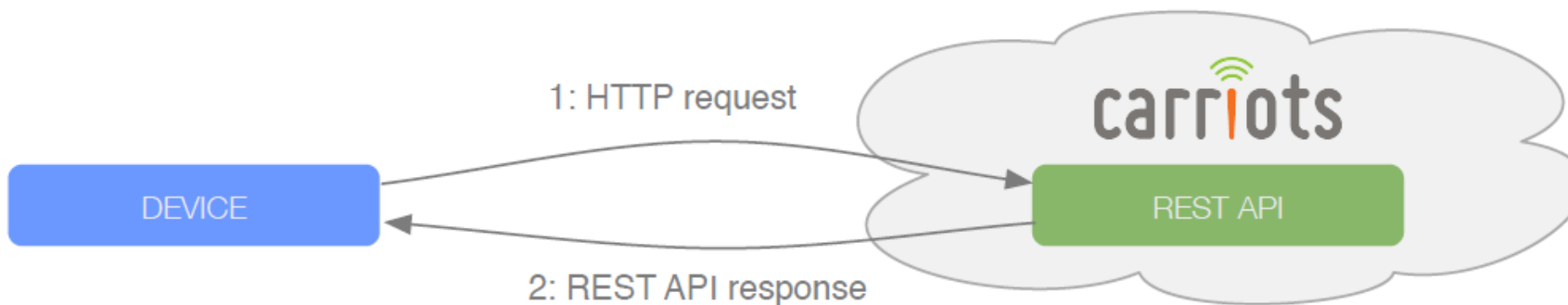
```
{
  "protocol": "v2",
  "at": "now",
  "device": "place_524@smartparking",
  "data": { "parking": "on" },
  "checksum": "2c0766329b4d4b3beb08...97ae7b7de2160be"
}
```

Checksum (HMAC) validation

Carriots automatic data timestamping

Device (Apikey must allow it)

Custom information to be sent



PLATAFORMAS

REST API

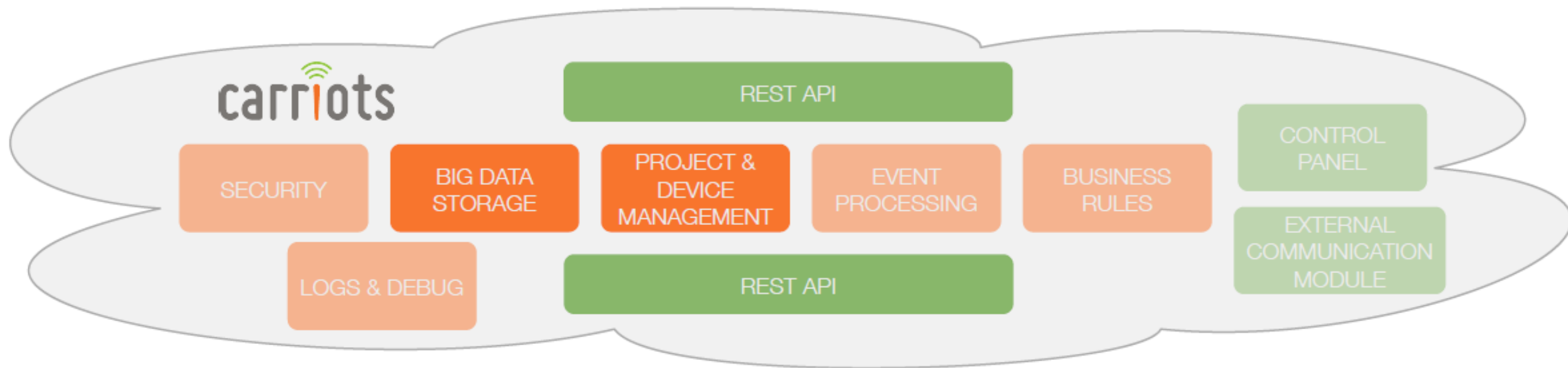
- Siguiendo un estándar del internet, Carriots implementa REST API sobre HTTPS para:
 - Recibir cantidades enormes de información del dispositivo.
 - Interactuar con las entidades de Carriots para crear panel de control, panel de instrumentos y herramientas de reporte.
- Ej.: Los sensores en el estacionamiento mandan el estatus (ocupado o vacío) de cada uno de los lugares a Carriots REST API con un query simple de HTTP. El panel de control utiliza REST API para la administración.

ALMACENAMIENTO DE BIG DATA

- Enorme cantidades de datos se almacenan en la arquitectura de Big Data en una estructura sin esquema.
- Proporciona a los proyectos de IoT flexibilidad para gestionar datos heterogéneos de diferentes dispositivos.

ADMINISTRACIÓN DE PROYECTO Y DISPOSITIVOS

- Los proyectos de IoT pueden organizarse para cumplir cualquier requerimiento. Carriots utiliza una simple lista de de pasos, ordenada jerárquicamente y ofrece la posibilidad de establecer el nivel de complejidad dependiendo de las necesidades del usuario.



REGLAS DE TRABAJO Y PROCESAMIENTO DE EVENTOS

- La lógica de un proyecto de IoT se lleva a cabo y se corre en la plataforma.
- Desde scripts sencillos a reglas de trabajo muy complejas se implementan en scripts de Groovy ejecutados en el motor SDK que son iniciados (y aislados) basados en eventos con un enfoque 'if-then-else'.
- Ej.:
if sensor.lugar = "ocupado" then estacionamiento.lugares -1
If estacionamiento.lugares=0 then mensaje="Estacionamiento lleno"

SEGURIDAD

- Los proyectos de IoT deben considerar la seguridad desde la fase del diseño.
 - **Apikeys** para definir los privilegios y la visibilidad.
 - **HTTPS** para encriptar las solicitudes y respuestas de REST API.
 - HMAC hash y contraseña previamente compartida para re-autenticar y verificar el contenido.
 - Encriptación y seguridad personalizada adicional.

LOGS Y DEPURACIÓN

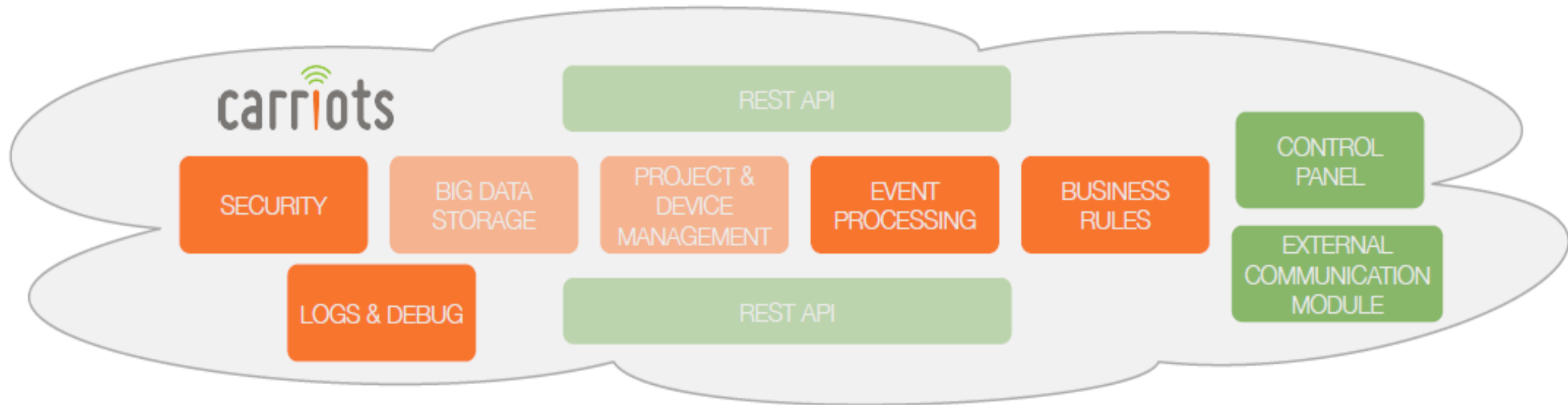
- Los mensajes de Log y la consola de depuración son herramientas para facilitar el desarrollo de un proyecto de IoT

PANEL DE CONTROL

- El panel de control de Carriots permite a todos los proyectos de IoT ser administrados por medio de una interface web, incluso si los paneles de control personalizados son desarrollados para los usuarios finales.

MÓDULO DE COMUNICACIÓN EXTERNA

- La plataforma de Carriots ofrece un poderoso módulo de comunicaciones que permite mandar emails, mensajes de texto, e interactuar con sistemas externos de integración.



EJEMPLO: lógica en Groovy que reacciona cuando se recibe información de un lugar de estacionamiento.

```
// Libraries
import com.carriots.sdk.Device;
import com.carriots.sdk.utils.BasicHttp;

// Free places counter update
def device = Device.find('ParkingControl@smartparking');
def places = new Integer(device.device_properties.FreePlaces);
device.device_properties.FreePlaces=places-1;
device.update();

// Place update
def place = Device.find(context.device);
place.device_properties['free']='no';
place.update();

// Display location
def led = Device.find('LedDisplay-12@smartparking');
def address = led.device_properties.address;
def token = led.device_properties.token;

// Display update
def basicHttp = new BasicHttp();
basicHttp.url = "http://" + address + "/message";
basicHttp.params=["text":"Free places: " + device.device_properties.FreePlaces,
"token":token];
basicHttp.send();
```

SDK LIBRARIES IMPORT

DATA BASE ACCESS

CUSTOM PROPERTIES MANAGEMENT

CARRIOTS -> DEVICE COMMUNICATION
IP address stored as device property

Panel de control de Carriots, ejemplo de creación de un módulo de escucha.

ENTITY SURVEILLANCE

EVENT TO WATCH

IF-THEN-ELSE APPROACH

GROOVY SCRIPTS

PRE-WRITTEN SCRIPTS

carriots®

Project Management

- Project
- Service

Device Management

- Groups
- Assets
- Devices
- Models

Data management

- Data streams
- Status streams
- Publish data
- Wizard Send Streams
- Data Export

Rules management

- Rules
- Listeners
- Wizard Create Listener
- Alarms

SMARTCITY

ADMINISTRATION

MY SETTINGS

DEBUG & LOG

HELP

ALARMS 30

LOGOUT

CARRIOTS CONTROL PANEL

Listener creation

Name

Description

Entity type

Project

Id

defaultProject@smartcity

Event

Event Data Received

If expression

1 context.data['parking']=='on'

Then expression

1 // Groovy expression

Then rule

Else expression

1 // Groovy expression

Else rule

Enabled

Create

List

FRONTEND

PANEL DE CONTROL DE CARRIOTS

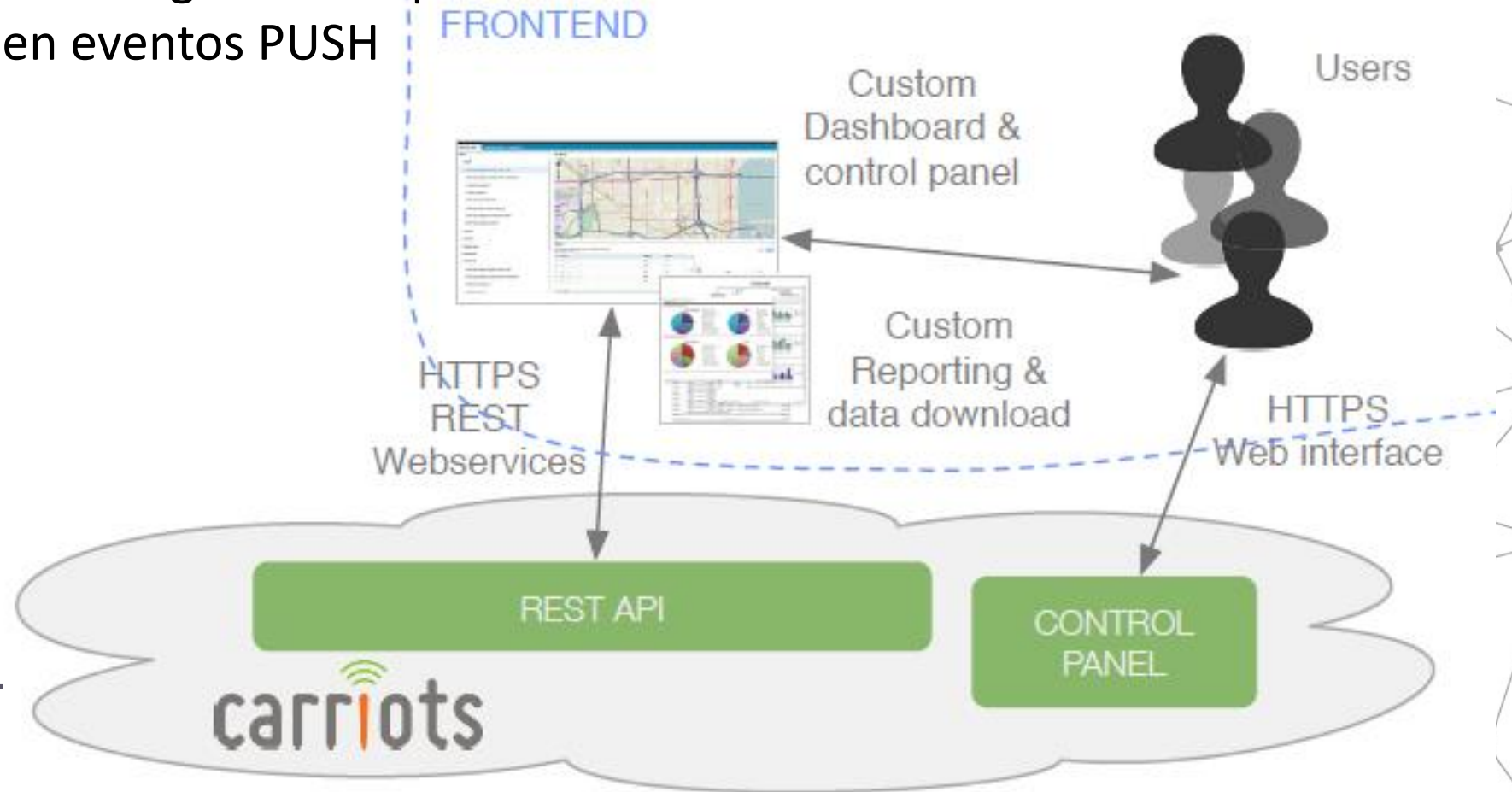
- El panel de control de Carriots es de las primeras herramientas necesarias para construir un proyecto de IoT. Esta herramienta permite: crear la jerarquía del proyecto para organizar las configuraciones particulares, administrar apikeys para definir privilegios y visibilidad, gestionar todos los datos del proyecto, utilizar herramientas de depuración, etc.
- Ej.: Definir los lugares de estacionamiento, establecer los mensajes de la pantalla, etc.

PANEL DEL TABLERO DE INSTRUMENTOS Y DE CONTROL

- Las interfaces que vé y utiliza el usuario final son cuadros de mando personalizados y paneles de control contruidos con Carriots REST API

MONITOREO

- Técnicas y herramientas de monitoreo:
 - Agrupamiento de REST API para la explotación de datos.
 - Descarga de archivos de datos.
 - Gráficas customizadas generadas por Carriots
 - Datos basados en eventos PUSH

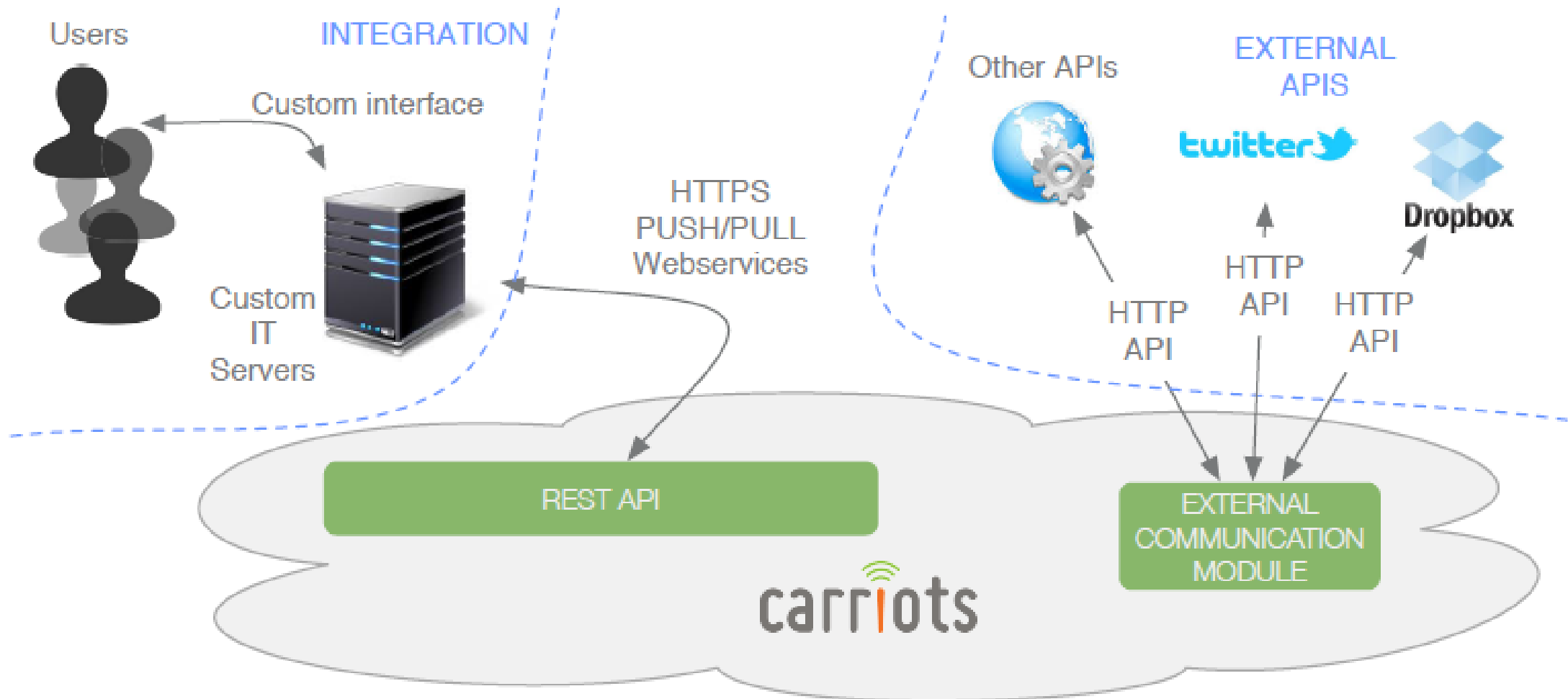


INTEGRACIÓN

BUILT-IN

- El motor SDK de Carriots cuenta con una serie de bibliotecas integradas para facilitar la integración:
 - Dropbox
 - Twitter
 - Mailing masivo
 - SMS Internacional
 - Sockets

INTEGRACIÓN



USANDO CARRIOTS

<https://www.carriots.com/>

Crear Cuenta

Panel de Control

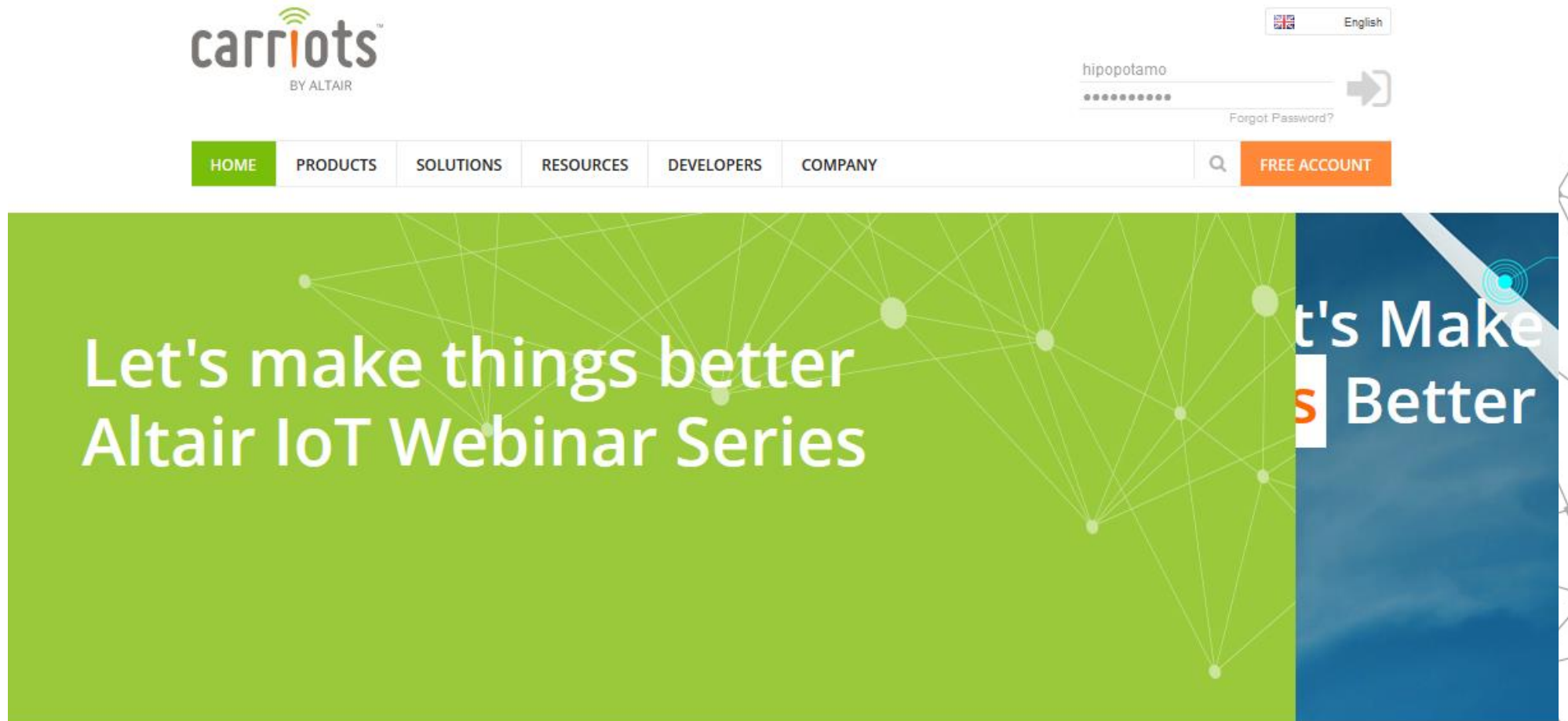
Envío y Recepción de Datos a Carriots

Listeners y Triggers

Integrando Arduino

Reportes

CREACIÓN DE CUENTA



The screenshot displays the website for 'carriots BY ALTAIR'. At the top left is the logo. To the right, there is a language selector set to 'English' and a login form with the username 'hipopotamo' and a masked password. Below the login form is a 'Forgot Password?' link. A navigation bar contains links for HOME, PRODUCTS, SOLUTIONS, RESOURCES, DEVELOPERS, and COMPANY, followed by a search icon and a 'FREE ACCOUNT' button. The main content area features a large green banner with the text 'Let's make things better Altair IoT Webinar Series' and a blue banner on the right with the text 'Let's Make Things Better'.

carriots[™]
BY ALTAIR

English

hipopotamo
.....

Forgot Password?

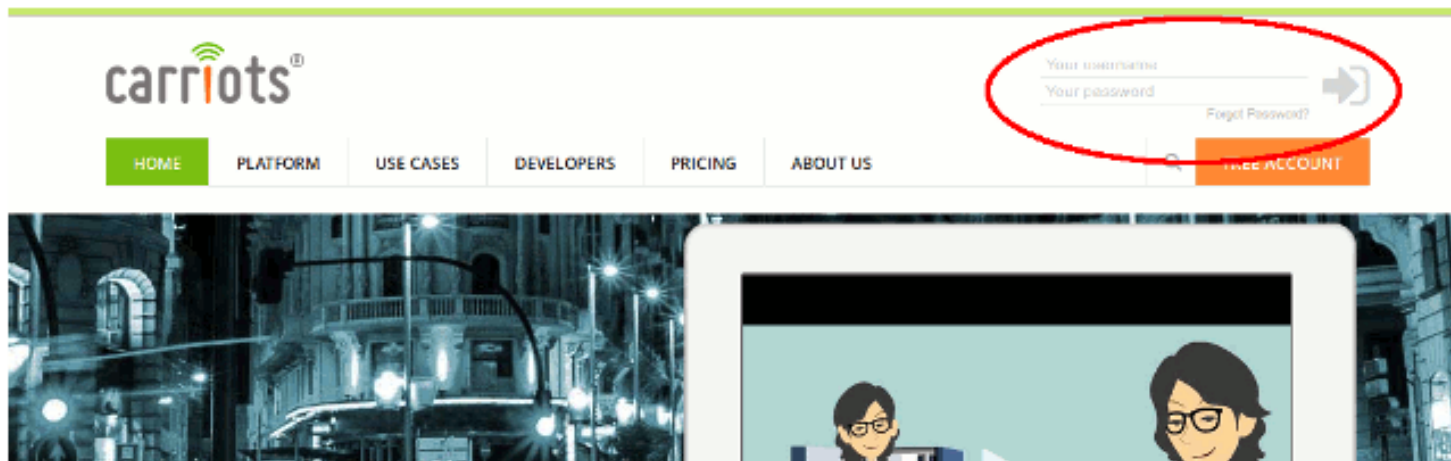
HOME PRODUCTS SOLUTIONS RESOURCES DEVELOPERS COMPANY

FREE ACCOUNT

Let's make things better
Altair IoT Webinar Series

Let's Make
Things Better

ACCESO



PANEL DE CONTROL







Bienvenido
HIPOPOTAMO
FREE

¡MEJORA TU CUENTA!



JERARQUÍA <

DATOS <

REGLAS <

CONFIGURACIÓN <

DASHBOARDS <

Bienvenido al Panel de Control de Carriots

Where the magic of IoT happens!

Dispositivos

El corazón de los proyectos IoT



3

Son el centro de todo proyecto IoT: las "Things" del "IoT". Cuando envías datos, siempre van asociados a un dispositivo.

Gestionar Dispositivos

Tramas

La información en los proyectos IoT



128

Las tramas de datos son la información que mantiene vivo tu proyecto IoT y siempre van asociadas a un dispositivo.

Gestionar Tramas

Listeners

"El ejecutor" de los proyectos IoT



3

Los Listeners comparan datos con expresiones y reglas y se ejecutan cuando el evento definido se produce.

Gestionar Listeners

Alarmas

Las notificaciones en los proyectos IoT



0

Son notificaciones creadas por el usuario o el propio sistema cuando algo falla o un dispositivo cambia de estado.

Gestionar Alarmas

LÍMITES / MINUTO 

Peticiones a la API (0 de 1000)

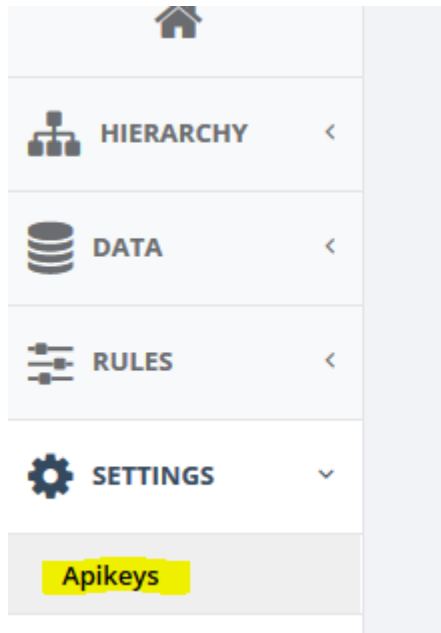
LÍMITES / DÍA 

Peticiones a la API (0 de 50000)

GEOLOCALIZACIÓN DE DISPOSITIVOS 

En este espacio deberías ver un mapa de tus dispositivos.

API KEY



Show 10 entries

Copy CSV PDF Print

Search

Apikey	Description	User	Default
[REDACTED]5cbfd4	Automatic Apikey Full	hipopotamo	✓
55 [REDACTED]490ab30ac	Automatic Apikey Stream On ly	hipopotamo	✗
[REDACTED]e9aea9567	Automatic Apikey Read Only	hipopotamo	✗

Showing 1 to 3 of 3 entries

DEVICES



Welcome
HIPOPOTAMO
FREE

UPGRADE ACCOUNT!



HIERARCHY ▾

Projects

Services

Groups

Assets

Devices

Models

Device List

Show 10 ▾ entries

Copy

CSV

PDF

Print

Name ▾	Description ▴	Time zone ▴	Created At ▴	Status ▴	Enabled ▴
SmartPhone	Celular que envia datos	America/Mexico_City	2017/05/10 23:17:48	disconnected	<input checked="" type="checkbox"/>
arduino	Arduino para demo IoT	America/Mexico_City	2017/07/01 17:36:32	disconnected	<input checked="" type="checkbox"/>
defaultDevice	defaultDevice from @hipopotamo.hipopotamo	Europe/Madrid	2015/07/09 13:15:09	disconnected	<input checked="" type="checkbox"/>

Showing 1 to 3 of 3 entries

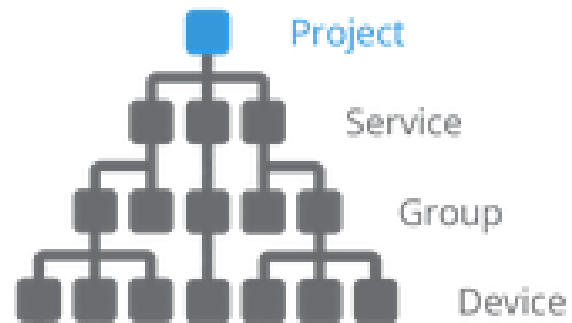
+ New Device

You can add 7 more devices

JERARQUÍAS

Hierarchy

You are here in the Carriots
Hierarchy



CREACIÓN DE DEVICE

Device Creation

Name

Type

Other

Checksum

Frequency Stream

1440

The time range in which a data stream should be received from this device, expressed in minutes

Latitude

Description

Sensor

Accelerometer

Time zone

Mexico City

Frequency Status

1440

The time range in which a status stream should be received from this device, expressed in minutes

Longitude

Geolocate your device

Enabled

ON

Networking

Group

Asset

ENVIAR DATOS (STREAM)

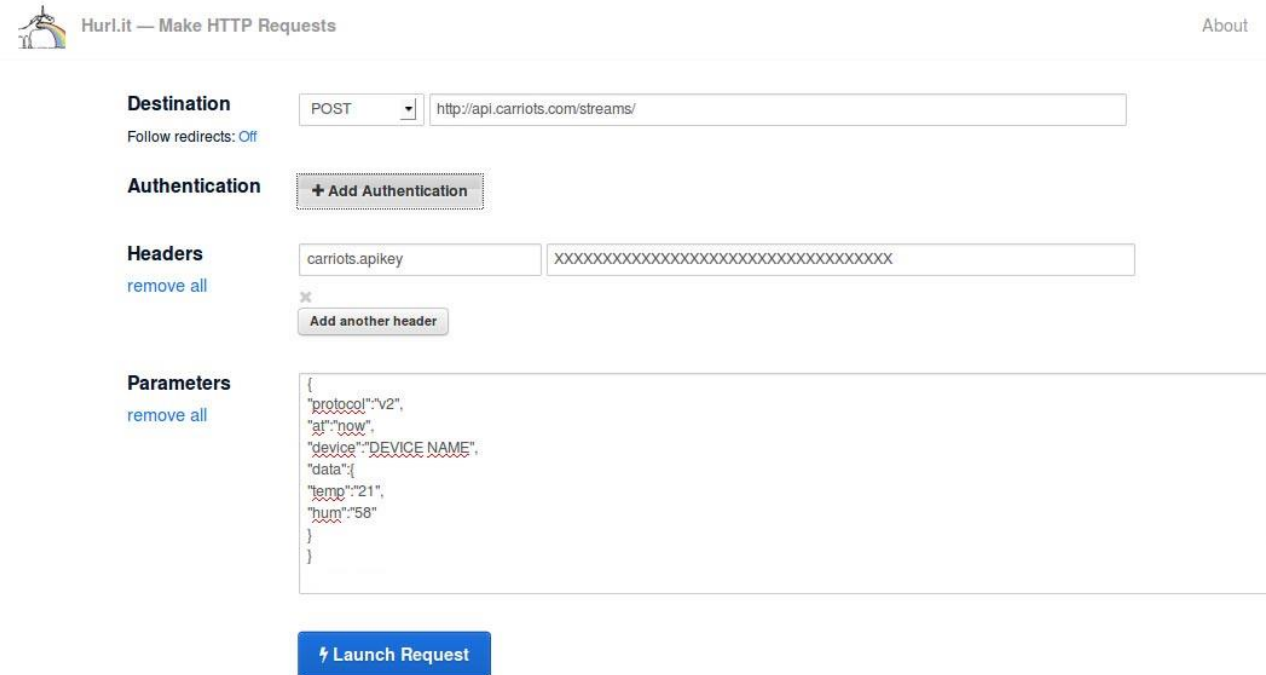
Payload: Información en formato Json o XML que contiene los datos

```
{  
  "protocol": "v2",  
  "at": 1355837673,  
  "device": "defaultDevice@cuenta.cuenta",  
  "data": {  
    "temp": 21,  
    "hum": 58  
  }  
}
```

```
<stream>  
  <protocol>v2</protocol>  
  <at>now</at>  
  <device>  
    defaultDevice@cuenta.cuenta  
  </device>  
  <data>  
    <temp>21</temp>  
    <hum>58</hum>  
  </data>  
</stream>
```

PRUEBA DE ENVIO DE DATOS

- HURL: <http://www.hurl.it/>
- Destination: POST,
`http://api.carriots.com/streams/`
- Headers: "carriots.apikey" y tu APIKEY
 - Si quieres enviar XML agregar "content-type" -> application/xml
- Parameters. Add Body
 - Escribe tu payload



The screenshot shows the Hurl.it web interface for making HTTP requests. The 'Destination' is set to POST at `http://api.carriots.com/streams/`. The 'Authentication' section has a button to '+ Add Authentication'. The 'Headers' section shows a header 'carriots.apikey' with a value of 'XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX'. There is a button to 'Add another header'. The 'Parameters' section shows a JSON payload:

```
{
  "protocol": "v2",
  "at": "now",
  "device": "DEVICE NAME",
  "data": {
    "temp": "21",
    "hum": "58"
  }
}
```

. At the bottom, there is a blue button labeled 'Launch Request'.

PRUEBA DE ENVIO DE DATOS

Poster: <https://addons.mozilla.org/en-us/firefox/addon/poster/>

PostMan: <https://www.getpostman.com/>

STREAMS



Welcome
HIPOPOTAMO
FREE

UPGRADE ACCOUNT!



HIERARCHY <



DATA v

Data Streams

Status Streams

Graph Widget

Data Streams

Carriots CPanel / Data streams



BYTES RECEIVED

25.65 K

Q Search

Data Streams List

Show 10 v entries

Copy

CSV

PDF

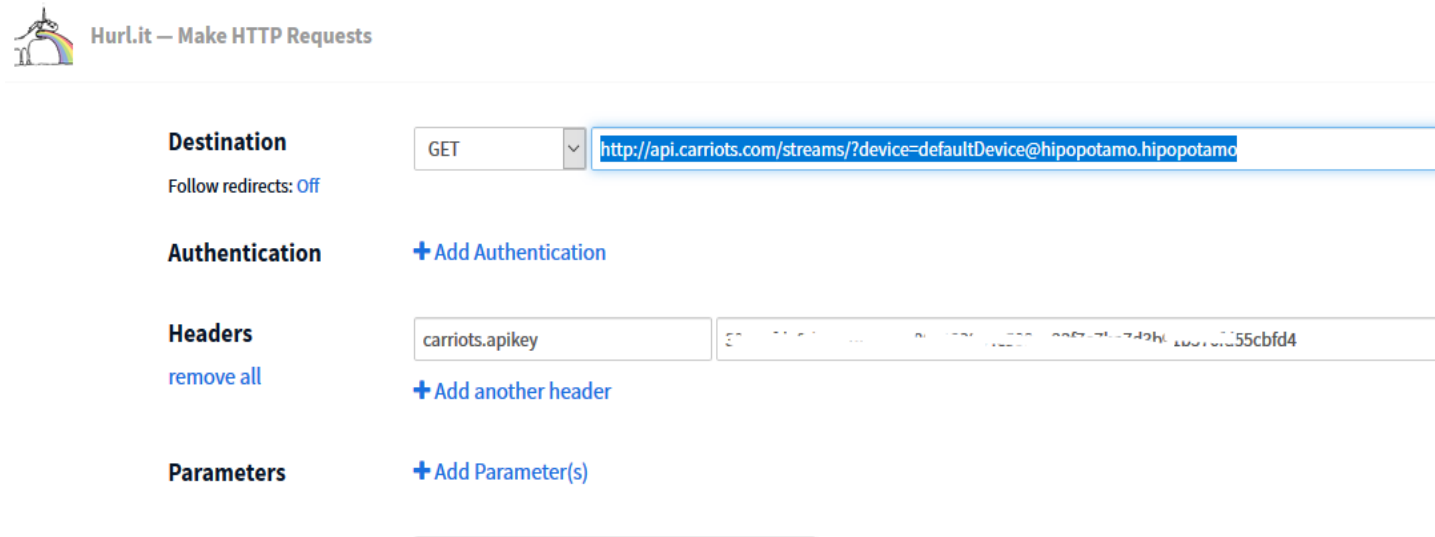
Print



	At	Device	Data
<input type="checkbox"/>	2017/11/01 00:45:36	arduino@hipopotamo.hipopotamo	{"latitud":"19.432568","temp":"56","longitud":"-99.132332"}
<input type="checkbox"/>	2017/11/01 00:45:21	arduino@hipopotamo.hipopotamo	{"latitud":"19.432011","temp":"56","longitud":"-99.133278"}
<input type="checkbox"/>	2017/10/31 23:41:38	arduino@hipopotamo.hipopotamo	{"latitud":"19.433319","temp":"56","longitud":"-99.135139"}

LECTURA DE DATOS

- Desde Hurl:
 - Destination: GET
`http://api.carriots.com/streams/?device=defaultDevice@cuenta.cuenta`
 - Headers: `carriots.apikey` , APIKEY



Hurl.it — Make HTTP Requests

Destination
Follow redirects: [Off](#)

GET `http://api.carriots.com/streams/?device=defaultDevice@hipopotamo.hipopotamo`

Authentication [+ Add Authentication](#)

Headers
[remove all](#)

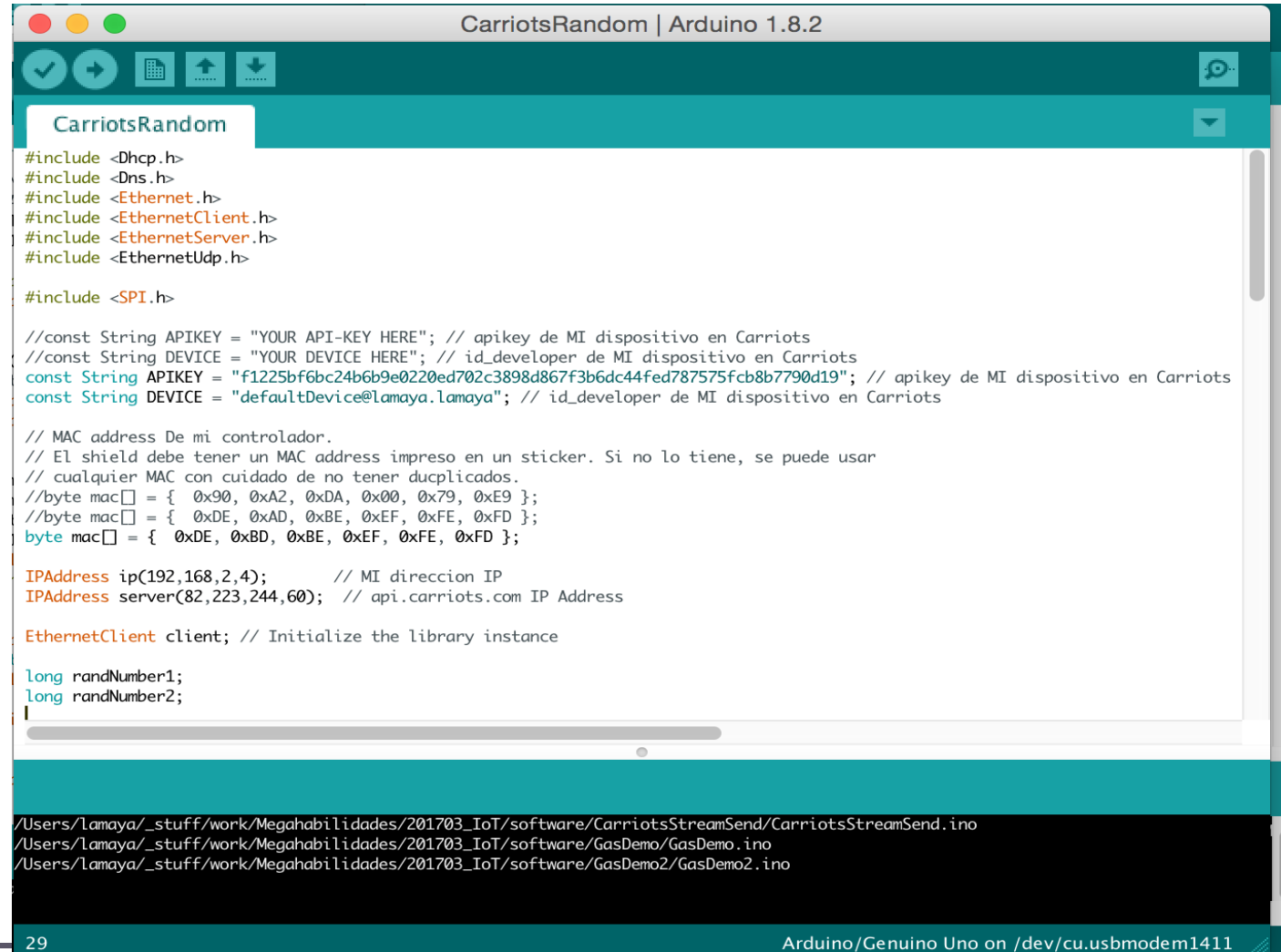
`carriots.apikey` `8f7771e7-7d2b-4b31-b055c5cbfd4` [+ Add another header](#)

Parameters [+ Add Parameter\(s\)](#)

ENVIO DE DATOS DESDE ARDUINO

- Construir el JSON igual que con “poster” o “postman”
- Tener cuidado con las comillas que van entre comillas
- Armar Payload y enviar
- Primer ejemplo sin sensor. Enviamos números aleatorios

ENVIO DE DATOS DESDE ARDUINO



```
CarriotsRandom | Arduino 1.8.2

CarriotsRandom

#include <Dhcp.h>
#include <Dns.h>
#include <Ethernet.h>
#include <EthernetClient.h>
#include <EthernetServer.h>
#include <EthernetUdp.h>

#include <SPI.h>

//const String APIKEY = "YOUR API-KEY HERE"; // apikey de MI dispositivo en Carriots
//const String DEVICE = "YOUR DEVICE HERE"; // id_developer de MI dispositivo en Carriots
const String APIKEY = "f1225bf6bc24b6b9e0220ed702c3898d867f3b6dc44fed787575fcb8b7790d19"; // apikey de MI dispositivo en Carriots
const String DEVICE = "defaultDevice@lamaya.lamaya"; // id_developer de MI dispositivo en Carriots

// MAC address De mi controlador.
// El shield debe tener un MAC address impreso en un sticker. Si no lo tiene, se puede usar
// cualquier MAC con cuidado de no tener ducplicados.
//byte mac[] = { 0x90, 0xA2, 0xDA, 0x00, 0x79, 0xE9 };
//byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xFD };
byte mac[] = { 0xDE, 0xBD, 0xBE, 0xEF, 0xFE, 0xFD };

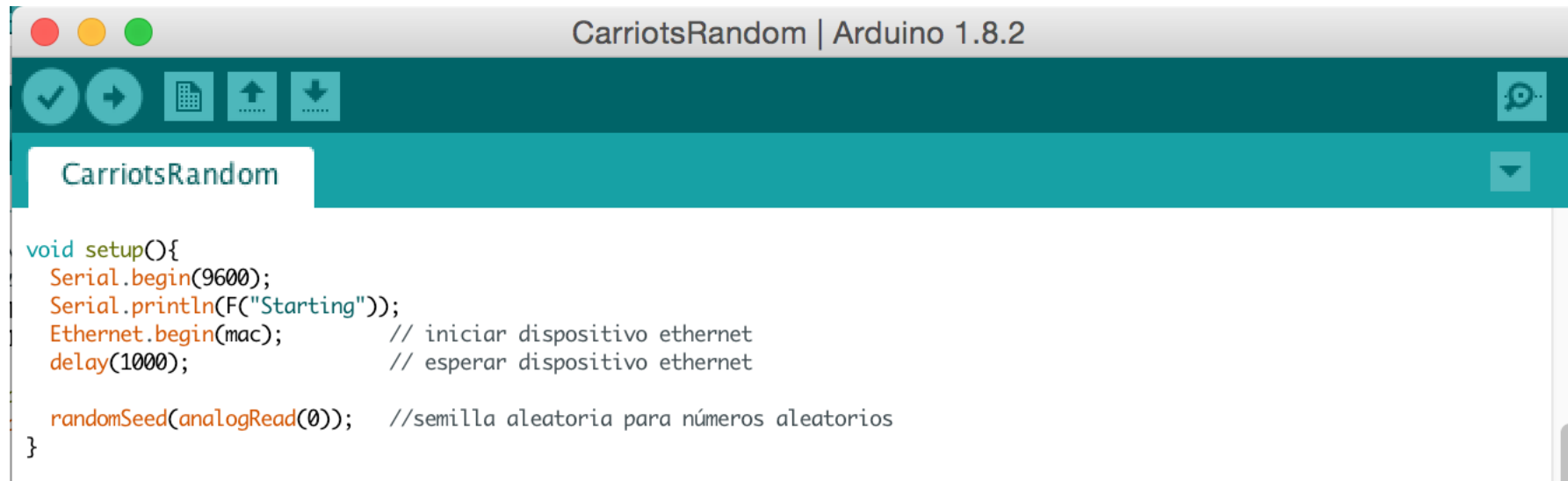
IPAddress ip(192,168,2,4); // MI direccion IP
IPAddress server(82,223,244,60); // api.carriots.com IP Address

EthernetClient client; // Initialize the library instance

long randomNumber1;
long randomNumber2;

/Users/lamaya/_stuff/work/Megahabilidades/201703_IoT/software/CarriotsStreamSend/CarriotsStreamSend.ino
/Users/lamaya/_stuff/work/Megahabilidades/201703_IoT/software/GasDemo/GasDemo.ino
/Users/lamaya/_stuff/work/Megahabilidades/201703_IoT/software/GasDemo2/GasDemo2.ino
```

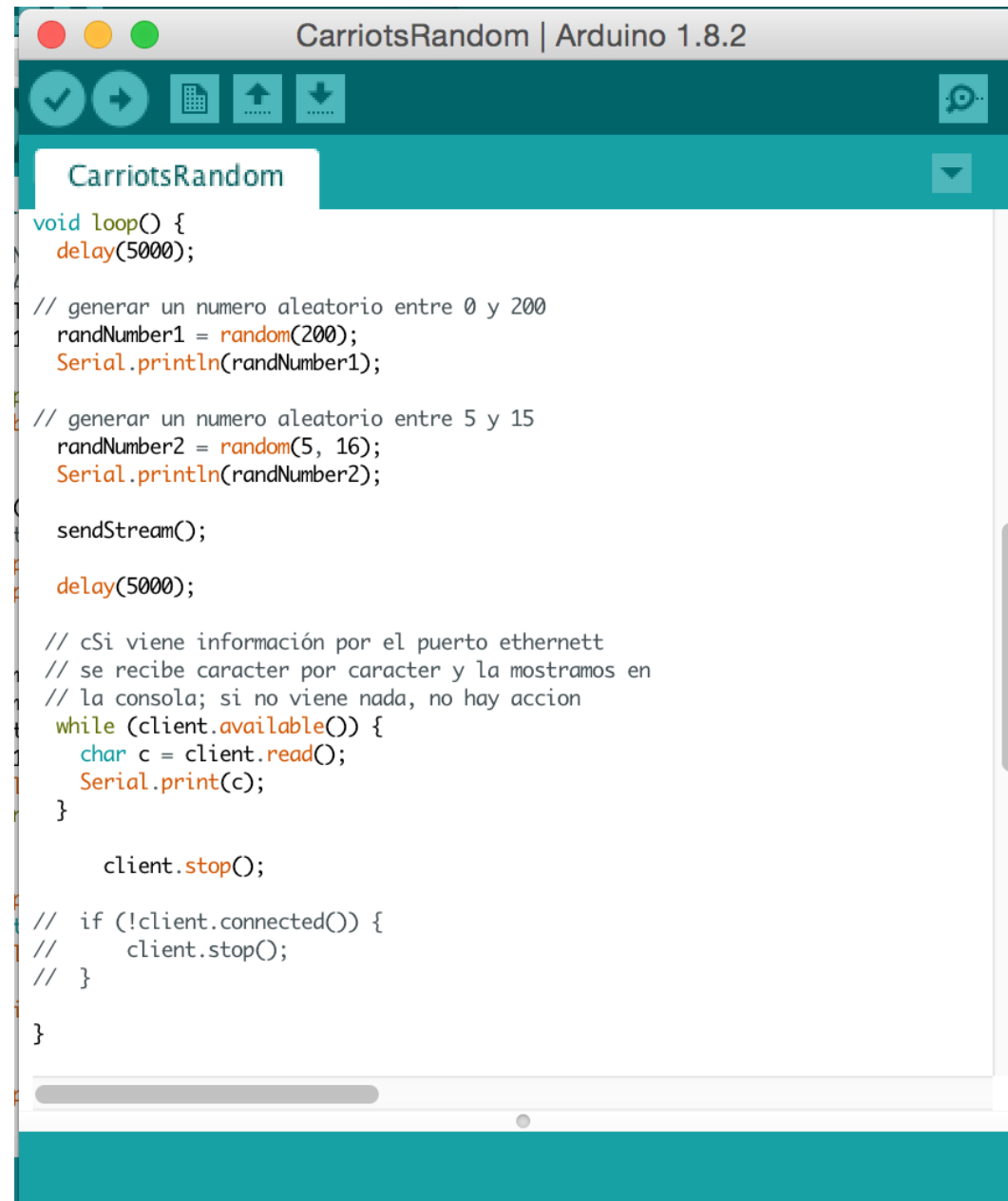
ENVIO DE DATOS DESDE ARDUINO



```
void setup(){
  Serial.begin(9600);
  Serial.println(F("Starting"));
  Ethernet.begin(mac);      // iniciar dispositivo ethernet
  delay(1000);              // esperar dispositivo ethernet

  randomSeed(analogRead(0)); //semilla aleatoria para números aleatorios
}
```

ENVIO DE DATOS DESDE ARDUINO

A screenshot of the Arduino IDE interface. The window title is 'CarriotsRandom | Arduino 1.8.2'. The toolbar at the top includes icons for checking, uploading, and downloading. The sketch editor shows the following code:

```
void loop() {  
  delay(5000);  
  
  // generar un numero aleatorio entre 0 y 200  
  randomNumber1 = random(200);  
  Serial.println(randomNumber1);  
  
  // generar un numero aleatorio entre 5 y 15  
  randomNumber2 = random(5, 16);  
  Serial.println(randomNumber2);  
  
  sendStream();  
  
  delay(5000);  
  
  // cSi viene información por el puerto ethernet  
  // se recibe caracter por caracter y la mostramos en  
  // la consola; si no viene nada, no hay accion  
  while (client.available()) {  
    char c = client.read();  
    Serial.print(c);  
  }  
  
  client.stop();  
  
  // if (!client.connected()) {  
  //   client.stop();  
  // }  
  
}
```

ENVIO DE DATOS DESDE ARDUINO



The screenshot shows the Arduino IDE interface with a window titled "CarriotsRandom | Arduino 1.8.2". The sketch is a C++ program designed to send data to the CarriotsRandom API. It includes a `sendStream()` function that builds a JSON string with fields for protocol, device, and a timestamp. The program attempts to connect to the server at port 80, prints the connection status, and then sends an HTTP POST request with the JSON data. Headers include Host, Accept, User-Agent, and Content-Type. The request body is the JSON string. The program also prints the length of the JSON string and the connection status. If the connection fails, it prints an error message.

```
// Send stream to Carriots
void sendStream()
{
  // Build the data field
  // String json = "{\"protocol\":\"v2\",\"device\":\""+DEVICE+"\",\"at\":\"no
  String json = "{\"protocol\":\"v2\",\"device\":\""+DEVICE+"\",\"at\":\"now\",
  Serial.println(json);

  if (client.connect(server, 80)) { // si logramos la conexion
    Serial.println(F("connectado"));

    // Make HTTP request
    client.println("POST /streams HTTP/1.1");
    client.println("Host: api.carriots.com");
    client.println("Accept: application/json");
    client.println("User-Agent: Arduino-Carriots");
    client.println("Content-Type: application/json");
    client.print("carriots.apikey: ");
    client.println(APIKEY);
    client.print("Content-Length: ");
    int thisLength = json.length();
    client.println(thisLength);
    client.println("Connection: close");
    client.println();

    client.println(json);
  }
  else {
    // si la conexion falló
    Serial.println(F("la conexion fallo"));
  }
}
```


ENVIO DE DATOS DESDE ARDUINO

```
String json =  
"{\"protocol\": \"v2\", \"device\": \"\"+DEVICE+\"\", \"at\": \"now\", \"data\": {\"  
rand0300\": \"\"+String(randNumber1)+\"\", \"rand515\": \"\"+String(randNu  
mber2)+\"\", \"randX\": \"\"+String((randNumber1+randNumber2)/2)+\"\"}}\";
```

PRACTICA

1. Crear un nuevo dispositivo (device) llamado “arduino”.
2. Utilizar **Hurl** o **Poster** para enviarle valores de Temperatura, Humedad, u otro
3. Verificar que los datos aparecen en el *stream*
4. Enviar el dato de temperatura o humedad desde el Arduino
5. Modificar la arquitectura del Arduino para enviar datos de un sensor al presionar un botón
6. Verificar que los datos aparecen en el stream

PROCESOS

- Listeners

Reacciona a un evento y en su caso dispara una acción

- Enviar un correo
- Escribir a una base de datos

- Triggers

Reacciona a un evento y envía datos de regreso a un dispositivo (push)

- Enviar una notificación
- Enviar un SMS
- Enviar un comando a un Arduino

LISTENER

Crear un nuevo Listener

The screenshot displays the 4X METRO web interface. On the left, a sidebar contains navigation options: 'HIERARCHY', 'DATA', 'RULES', and 'Listeners' (which is highlighted with a yellow background). The main content area shows a table with the following data:

Listener Name	Rule	Event Data	Destination
ZohoReporter	Enviar temperaturas a Zoho	Event Data Received	arduino@hipo)

Below the table, it says 'Showing 1 to 3 of 3 entries'. A green button with a white plus sign and the text '+ New Listener' is visible. At the bottom of the main area, there are two tabs: 'Rules' and 'Alarms'.

PROGRAMAR EL LISTENER

If Expression:

`context.data.temp>70`

Then Expression:

```
import com.carriots.sdk.utils.Email;  
import com.carriots.sdk.utils.BasicHttp;  
  
// Enviar un correo si el parámetro esta fuera de rango  
def email = new Email ();  
email.to="<your email>";  
email.subject="Alerta de tu dispositivo";  
email.message="Valor excedido: "+context.data.temp;  
email.send();
```

Listener Creation

Name

AlertaCorreo

Description

Enviar una alerta si un valor es muy alto

Entity type

Device

Event

Event Data Received

Id

arduino@hipopotamo.hipopotamo

If expression

1 context.data.temp>70

Then
expression

```
1 // Enviar un correo si el parámetro esta fuera de rang
2 def email = new Email ();
3 email.to="f . . @live.com";
4 email.subject="Alerta de tu dispositivo";
5 email.message="Valor excedido: "+context.data.temp;
6 email.send();
7
```

Then rule

Else
expression

1

Else rule

Enabled

ON

Create Listener

Back to List

EJEMPLO: ENCENDER O APAGAR UN ELEMENTO EN UN DISPOSITIVO

If:

```
context.data.boton=="on"
```

Then:

```
import com.carriots.sdk.utils.Email;
```

```
import com.carriots.sdk.utils.BasicHttp;
```

```
// enviar un mensaje a un webservice en el dispositivo
```

```
// se envía un token para demostrar mis credenciales
```

```
def basicHttp = new BasicHttp();
```

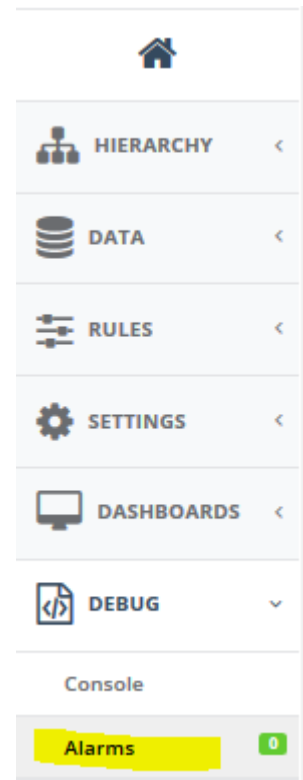
```
basicHttp.url ="http://control.arduino:9090/enciende?token=pa$$wOrd";
```

```
basicHttp.verb ="GET";
```

```
basicHttp.send();
```

VERIFICAR LA VALIDEZ DEL CÓDIGO

Verifica si existe una alarma en el menú de depuración

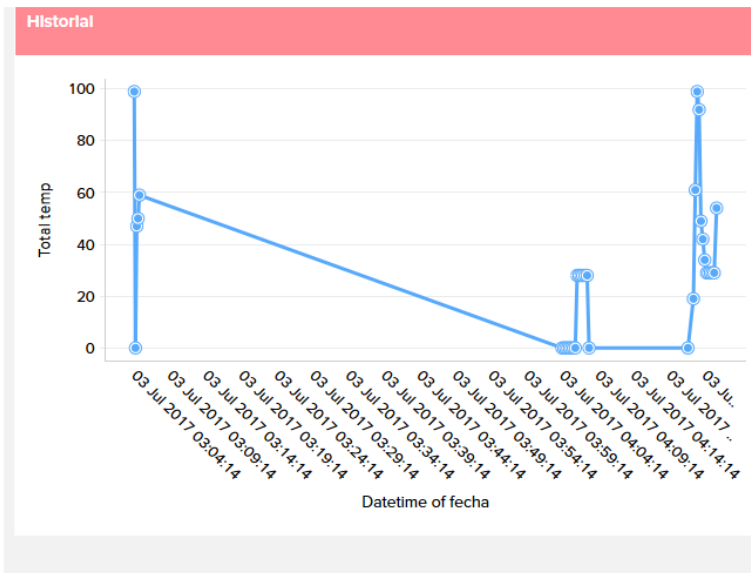


VISUALIZACIÓN DE DATOS

Crearemos un **Listener** que envíe los datos a una aplicación de reportes.

Monitoreo de Temperaturas en la ciudad

Tablero de Control



Promedio

	Date of fecha ↓	Avg temp
1	03 Jul, 2017	29.94118

Lecturas Recibidas

	fecha	↑ .#	temp	.#	latitud	.#	longitud
1	03 Jul, 2017 04:25:53			54	19.432011		-99.1332
2	03 Jul, 2017 04:25:33			29	19.432077		-99.1335
3	03 Jul, 2017 04:25:17			29	19.433475		-99.1341
4	03 Jul, 2017 04:25:01			29	19.435165		-99.1339
5	03 Jul, 2017 04:24:45			29	19.435692		-99.1380
6	03 Jul, 2017 04:24:29			29	19.436146		-99.1408
7	03 Jul, 2017 04:24:13			34	19.435093		-99.1420
8	03 Jul, 2017 04:23:57			42	19.434149		-99.1408
9	03 Jul, 2017 04:23:41			49	19.433746		-99.1381

APP DE REPORTES

Zoho reports

<http://reports.zoho.com>



Create your Zoho Reports Account in less than 60 seconds

Name

Company

Phone

Zoho Username

Email Address

Password

Re-enter Password

Image Text: Enter the text you see in the below image:

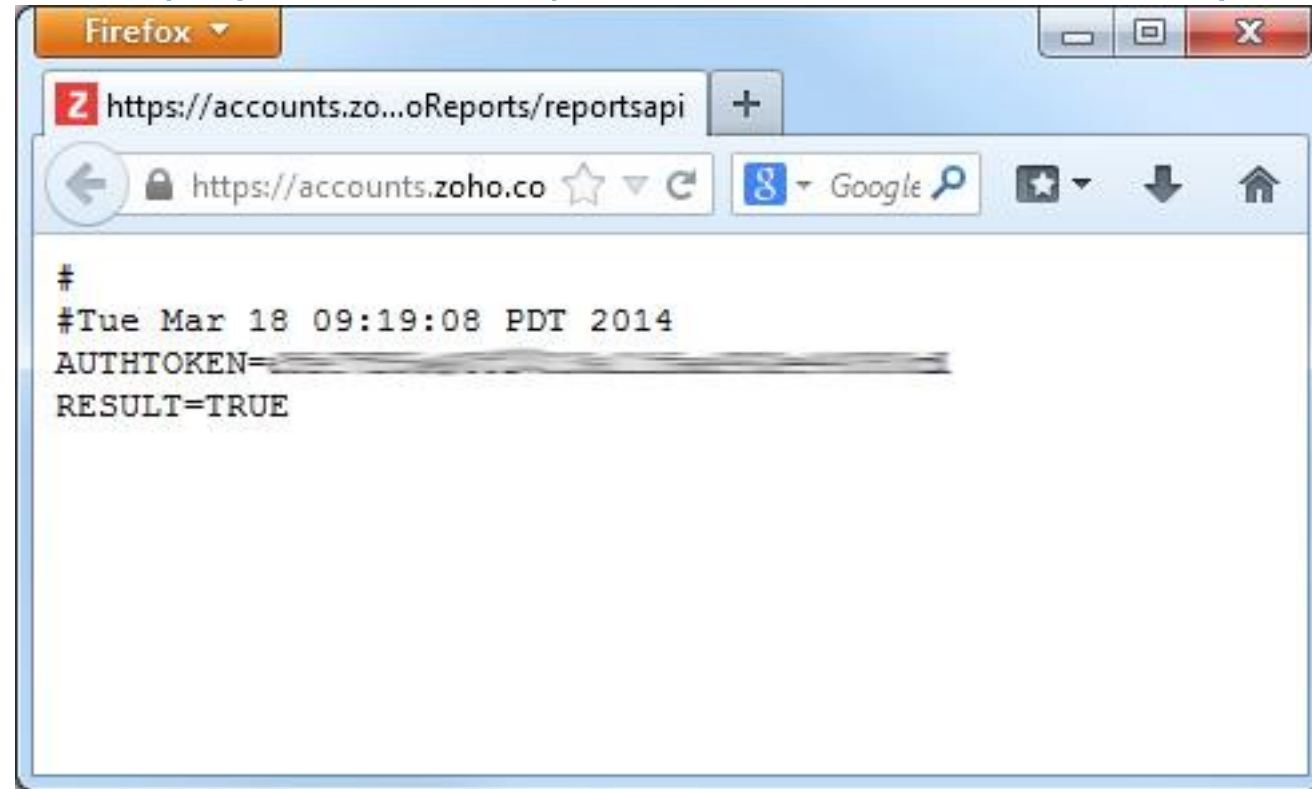
☐ I agree to the [Terms of Service](#) and [Privacy Policy](#)

☐ Yes, Subscribe me to Zoho Newsletter (Optional)

Sign Up

OBTENER TOKEN

<https://accounts.zoho.com/apiauthtoken/create?SCOPE=ZohoReports/reportsapi>



CREAR BASE DE DATOS PARA REPORTES

Import Your Data

Import your data from a wide variety of sources to create insightful reports & dashboards.

Getting Started

Recent Items



Files & Feeds



Cloud Storage/Drive



Cloud Databases



Local Databases



Zoho CRM



Salesforce CRM



My Databases

Shared Databases

Search Databases



Templates

Blank Database

Create a blank reporting database. This can be created later.



Blank Database



Create a Blank Reporting Database

Create a blank reporting database. Tables can be created later.

Database Name 

Carriots

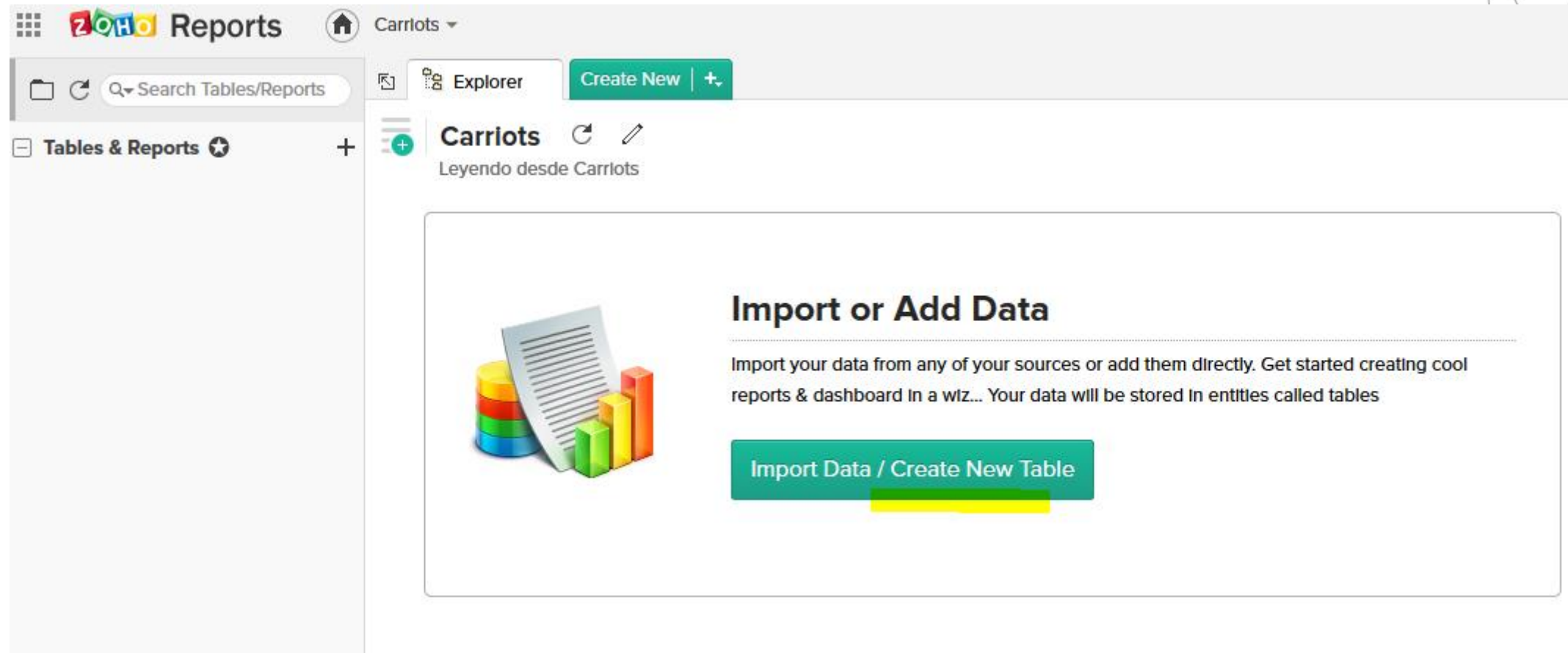
Description

Leyendo desde Carriots

Create

Cancel

CREAR UNA TABLA



The screenshot displays the Zoho Reports web application interface. At the top, the 'Zoho Reports' logo is visible alongside a home icon and a 'Carriots' dropdown menu. Below this, a search bar labeled 'Search Tables/Reports' is present. The left sidebar features a 'Tables & Reports' section with a plus icon. The main content area is titled 'Carriots' with a subtitle 'Leyendo desde Carriots'. A prominent green button labeled 'Create New' is located in the top right of the main area. The central focus is a large box titled 'Import or Add Data' which includes an illustration of a document and a bar chart. The text within this box instructs users to import data from various sources or add it directly to create reports and dashboards. A green button at the bottom of this box is labeled 'Import Data / Create New Table'.

Zoho Reports Carriots

Search Tables/Reports

Tables & Reports

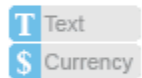
Carriots
Leyendo desde Carriots

Create New

Import or Add Data

Import your data from any of your sources or add them directly. Get started creating cool reports & dashboard in a wiz... Your data will be stored in entities called tables

Import Data / Create New Table



Using Design View

Create a table by specifying the column names, data types and column properties

Carriots ▾



Explorer



Edit Design*×

Create New



Edit Design: Untitled-1



Save



Refresh



Add Column

	Column Name	Data Type	Mandatory	
*1	fecha	Date	No	
*2	temperatura	Number	No	
*3	humedad	Number	No	
*		Plain Text	No	

Save As



Name:

Sensores

Save In Folder:

Tables & Reports



Description:

OK

Cancel

CREAR LISTENER CARRIOTS

THEN:

```
import com.carriots.sdk.utils.BasicHttp;
```

```
import groovy.json.JsonBuilder;
```

```
long d=Long.parseLong(context.envelope.at.toString())*1000; //extraer la fecha del payload
```

```
String date = new java.text.SimpleDateFormat("MM/dd/yyyy HH:mm:ss").format(new java.util.Date (d));
```

```
//java.text.SimpleDateFormat isoFormat = new java.text.SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
```

```
//isoFormat.setTimeZone(TimeZone.getTimeZone("UTC-6"));
```

```
//Date date1 = isoFormat.parse(date);
```

```
def request= new BasicHttp();
```

```
//Las siguientes tres líneas son un solo renglón!
```

```
request.url = "https://reportsapi.zoho.com/api/userZoho@dominio/Carriots/Sensores?
```

```
ZOHO_ACTION=ADDROW&ZOHO_OUTPUT_FORMAT=JSON&ZOHO_ERROR_FORMAT=JSON&
```

```
authtoken=8ff84cb57d90924b5d90c8078b8cdff2&ZOHO_API_VERSION=1.0";
```

```
request.verb = "POST";
```

```
request.headers = ["Content-Type": "application/x-www-form-urlencoded"];
```

```
request.payload= "temperatura="+context.data.temp+
```

```
"&humedad="+context.data.hum+
```

```
"&fecha="+date;
```

```
request.send();
```

IF:
true



If expression

```
1. true
```

Then expression

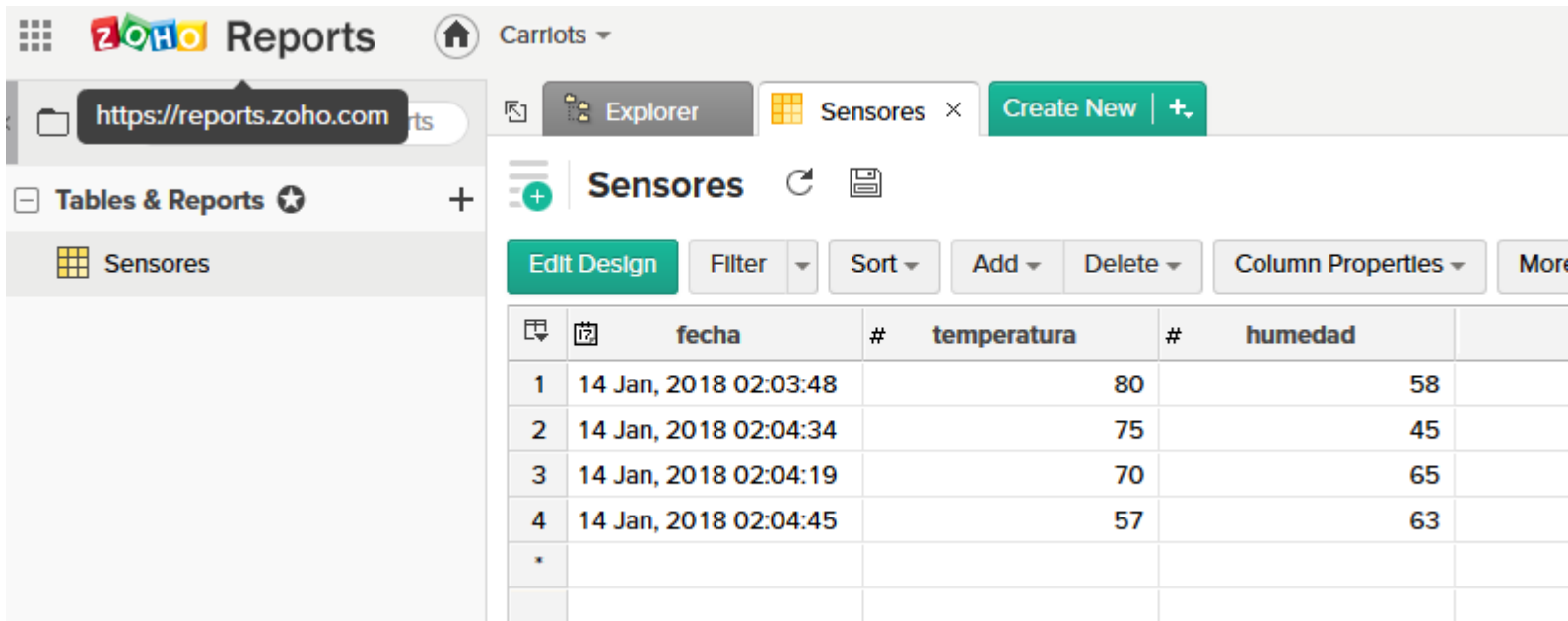
```

1. import com.carriots.sdk.utils.BasicHttp;
2. import groovy.json.JsonBuilder;
3. long d=Long.parseLong(context.envelope.at.toString())*1000; //extraer la fecha del payload
4. String date = new java.text.SimpleDateFormat("MM/dd/yyyy HH:mm:ss").format(new java.util.Date (d));
5. //java.text.SimpleDateFormat isoFormat = new java.text.SimpleDateFormat("MM/dd/yyyy HH:mm:ss");
6. //isoFormat.setTimeZone(TimeZone.getTimeZone("UTC-6"));
7. //Date date1 = isoFormat.parse(date);
8.
9. def request= new BasicHttp();
10. request.url = "https://reportsapi.zoho.com/api/rtrejo@megahabilidades.mx/Carriots/Sensores?ZOHO_ACTION=ADDROW&ZOHO_OUTPUT_FORMAT=JSON&ZOHO_ERROR_FORMAT=JSON&authtoken=8ff84151-521b-4211-b278b8c1f2&ZOHO_API_VERSION=1.0";
11. request.verb = "POST";
12. request.headers = ["Content-Type": "application/x-www-form-urlencoded"];
13. request.payload= "temperatura="+context.data.temp+
14. "&humedad="+context.data.hum+
15. "&fecha="+date;
16. request.send();

```

ENVÍO DE DATOS

Envía datos de temperatura y humedad usando Arduino o Hurl.
(Es probable que debas refrescar la vista de la tabla en Zoho)



The screenshot shows the Zoho Reports web interface. The browser address bar displays `https://reports.zoho.com`. The left sidebar contains a 'Tables & Reports' section with a 'Sensores' table listed. The main content area is titled 'Sensores' and includes a table with the following data:

	fecha	#	temperatura	#	humedad
1	14 Jan, 2018 02:03:48		80		58
2	14 Jan, 2018 02:04:34		75		45
3	14 Jan, 2018 02:04:19		70		65
4	14 Jan, 2018 02:04:45		57		63
*					

CREACIÓN DE UN REPORTE

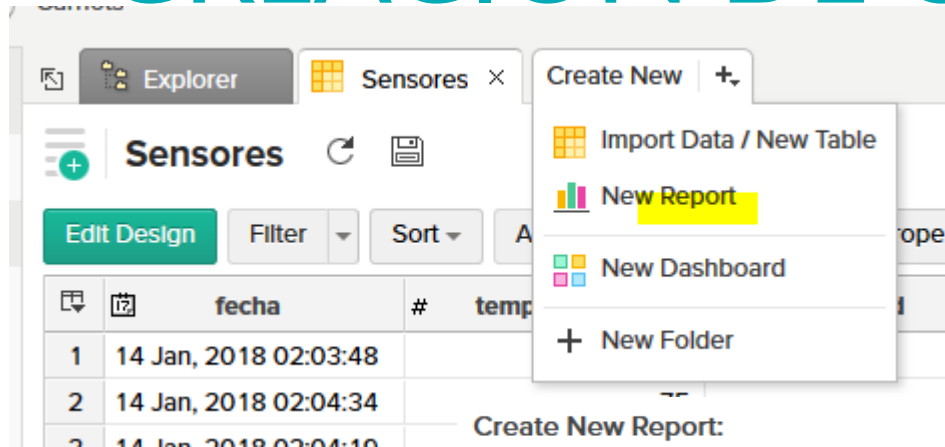


Chart View

Chart View allows you to create a graphical view using different types of charts.



Tabular View

Tabular View provides you a way to list your values in a simple tabular format.



Pivot View (Matrix View)

Pivot View allows you to view data summarized in a grid both in horizontal and vertical columns.



Summary View

Summary View allows you to view data with grouping and data summaries.

Auto Generate Reports



Similar to Another Table

Will generate reports for the selected table, similar to what is available for another table in this database.






With Auto Analysis

Will auto generate reports for the selected table, by analyzing & applying advanced rules on the data in the table.

Select/Drag and Drop the Columns






Sensores

- ☒  fecha
- ☒ # temperatura
- ☒ # humedad

Untitled-1   Save

View Mode

Sort

More Charts

Merge Axis

Graph

Filters (0)

User Filters

Reset All

X-Axis: fecha Date&time X

Y-Axis: temperatura Actual(M) X

humedad Actual(M) X

Color: Drop your column here

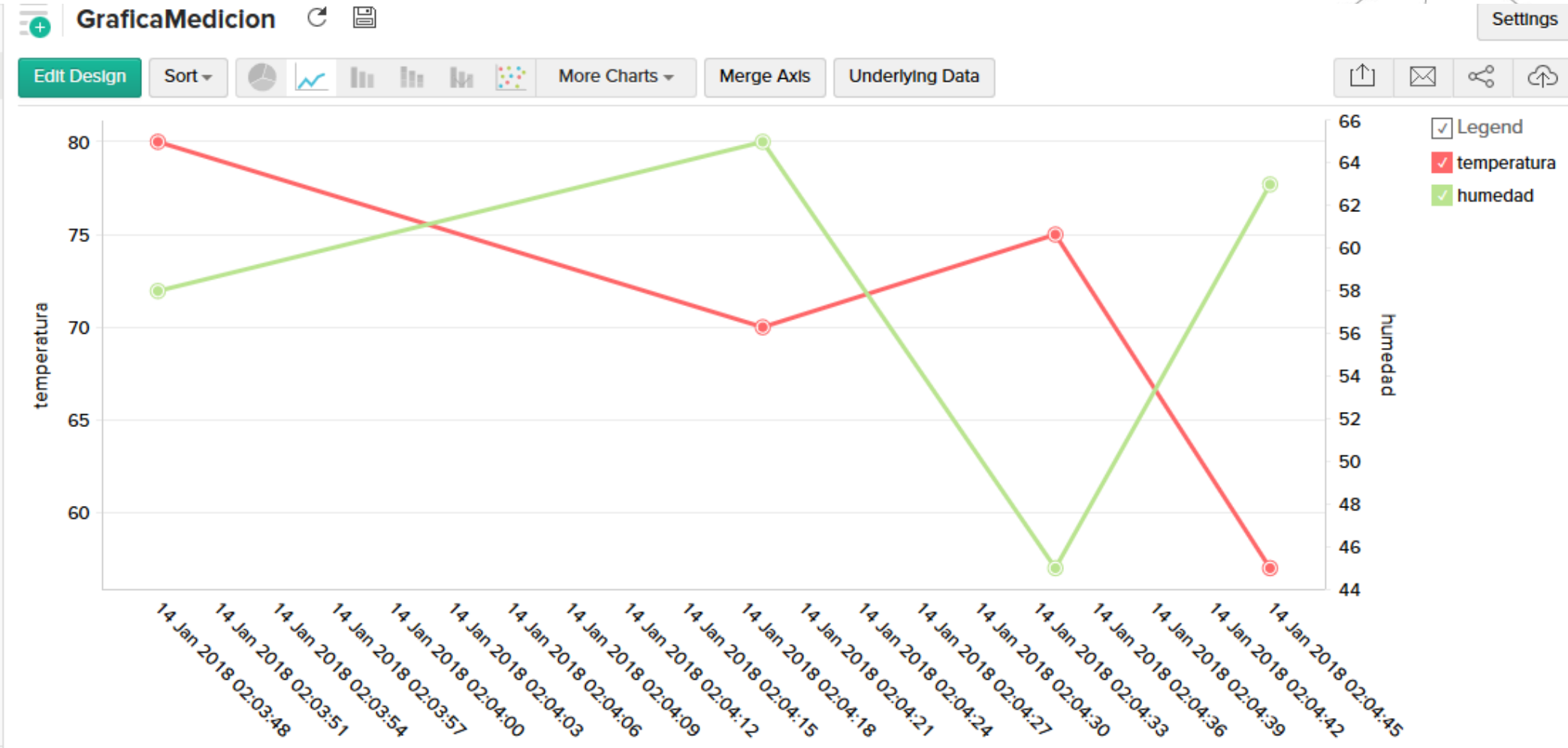
Text: Drop your column here

Size: Drop your column here

Tables & Reports ☆

GraficaMedicion

Sensores



REPORTE DE AGREGACIÓN

Crear un nuevo reporte, tipo Summary

Create New Report:



Chart View

Chart View allows you to create a graphical view using different types of charts.



Pivot View (Matrix View)

Pivot View allows you to view data summarized in a grid both in horizontal and vertical columns.



Tabular View

Tabular View provides you a way to list your values in a simple tabular format.



Summary View

Summary View allows you to view data with grouping and data summaries.

View Mode

Underlying Data

Themes

Summary

Filters (0)

User Filters

[Reset All](#)

Group by:

fecha

Mon&Year

X

Summarize:

temperatura


Avg

X

humedad

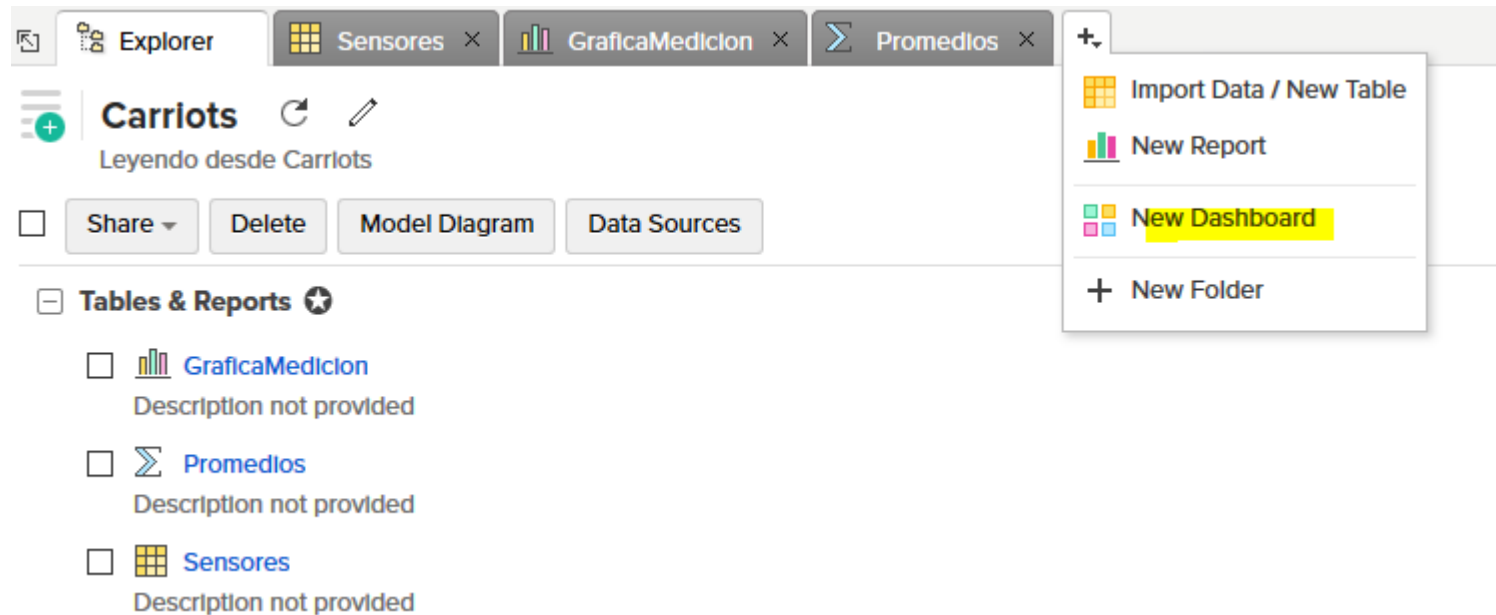
Avg

X

 [Click Here to Generate Summary](#)


	Month&Year of fecha ↓	Avg temperatura	Avg humedad
1	Jan 2018	70.50	57.75

CREACION DE UN DASHBOARD




Arrastrar los reportes y acomodar en el dashboard.
Crear filtros si se desea

View Mode

 Add Widget

T Add Text

 Add Image



User Filters

☒ Auto Add User Filters

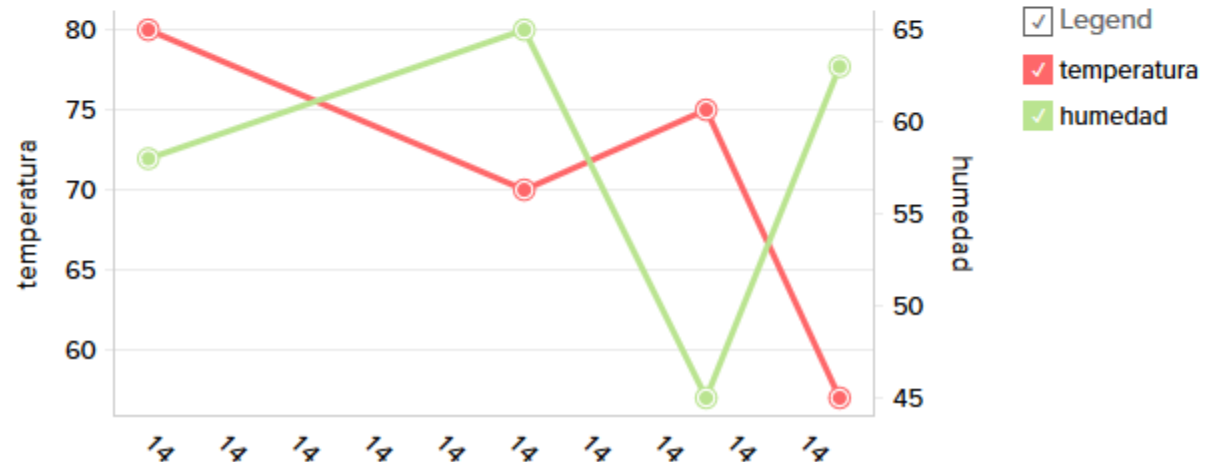
fecha:

— Select —



+ Add User Filters

GraficaMediclon



Promedios

	Month&Year of fecha ↓	Avg temperatura
1	Jan 2018	70.50
	< >	

Edit Design

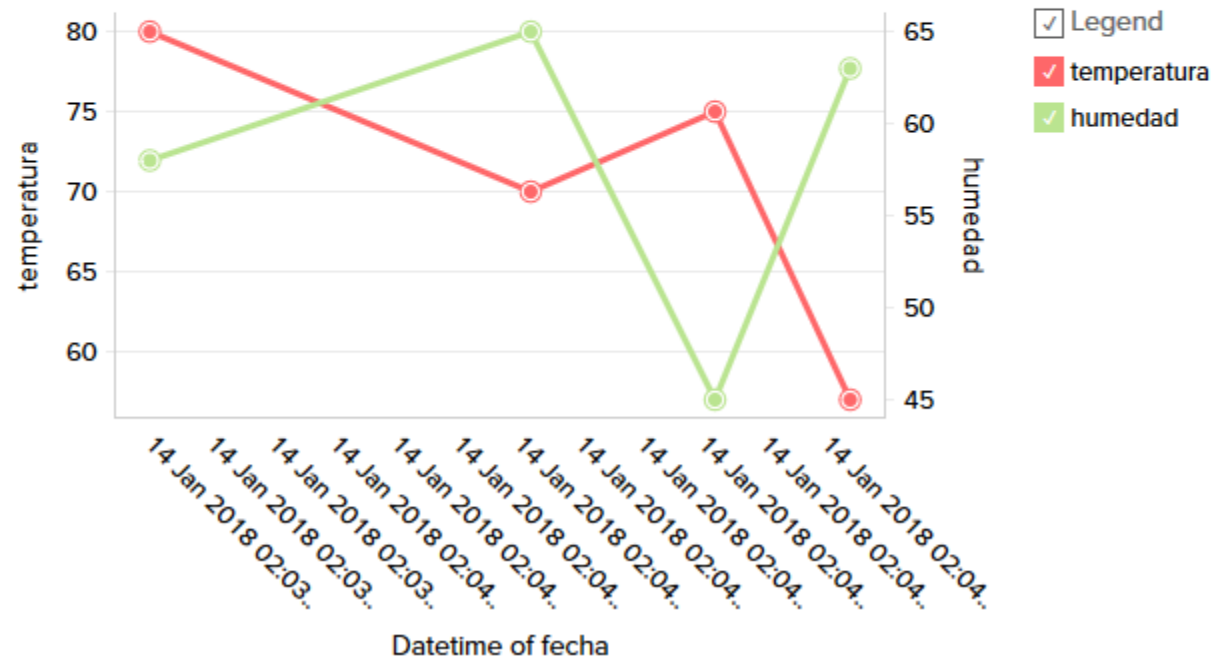
Themes







fecha:

— Select —

GraficaMediclon



MOSTRAR EN WEB



Embed In Website/Blog

URL / Permalink for this View

Create Slideshow

Manage Slideshows

Embed Snippet for 'Dashboard'




Access Permission: ☒ Access with Login  ☐ Access without Login 

Copy/Paste below html snippet into your web page:

```
<iframe frameborder=0 width="800" height="600" src="https://reports.zoho.com/open-view/12100...86"></iframe>
```

Embed & URL Options:

Set Screen Size: Width: Height:

☒ Auto Refresh every   secs 

Include:

☐ Toolbar ☒ Title ☒ Description

> Specify Filter Criteria 

Close

MOSTRAR EN WEB

```
<!DOCTYPE html>
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
  <head> <meta charset="utf-8" />
    <title>DashBoard</title>
  </head>
  <body>
    <h1> Reporte de Sensores</h1>
    <iframe frameborder=0 width=100% height="800"
src="https://reports.zoho.com/ZDBDataSheetView.cc?OBJID=124371900xx00004092&STAN
DALONE=true&WIDTH=800&HEIGHT=600&INTERVAL=120&REMTOOLBAR=true&INCLU
DETITLE=true&INCLUDEDESC=true">
    </iframe>
  </body>
</html>
```


IFTTTY GOOGLE DOCS

SDK

<https://www.carriots.com/documentation/sdk>

Incluye herramientas para:

- Email
- Http
- SMS
- Twitter
- MQTPP
- Contexto de datos de Carriots

4X METRO

CENTRO DE INNOVACIÓN INDUSTRIAL

<http://cii4xmetro.com/>

 www.facebook.com/CII4xMetro

 [@CII4xMetro](https://twitter.com/CII4xMetro)

 contacto@cii4xmetro.com