ARQUITECTURAS DE NUBE PARA Internet of Things

REDTPI4.0

CYTED

PROGRAMA IBEROAMERICANO DE CIENCIA
Y TECNOLOGÍA PARA EL DESARROLLO

Agenda

Sesión 1: 2 horas sincrónicas + 4 horas independientes

1. Introducción a IoT y arquitecturas
2. Introducción y principios de Nube
3. Manejo del laboratorio
4. Ejercicios de introducción (trabajo independiente)

Sesión 2: 2 horas sincrónicas + 4 horas independientes

1. Arquitecturas de Nube – Generalidades
2. El Agente/Orquestador/Broker
3. Sistemas de almacenamiento
4. Sistemas de ETL
5. Sistemas de Toma de decisiones
6. Sistemas de Visualización
7. Ejercicios de Agentes y escritura de datos (trabajo independiente)

Sesión 3: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de Almacenamiento – Modos de almacenamiento/arquitecturas con ventajas y desventajas

Sesión 4: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de ETL
2. Visualización de datos
3. Taller de ETL y visualización de datos – Ejercicio PM2.5 (trabajo independiente)

Sesión 5: 2 horas sincrónicas + 4 horas independiente

1. Sistemas de toma de decisiones
2. Interfaces de usuario y desarrollo de apps
3. Taller de toma de decisiones y desarrollo de apps (preventivos y reactivos)

Sesión 6: 2 horas sincrónicas + 4 horas independiente

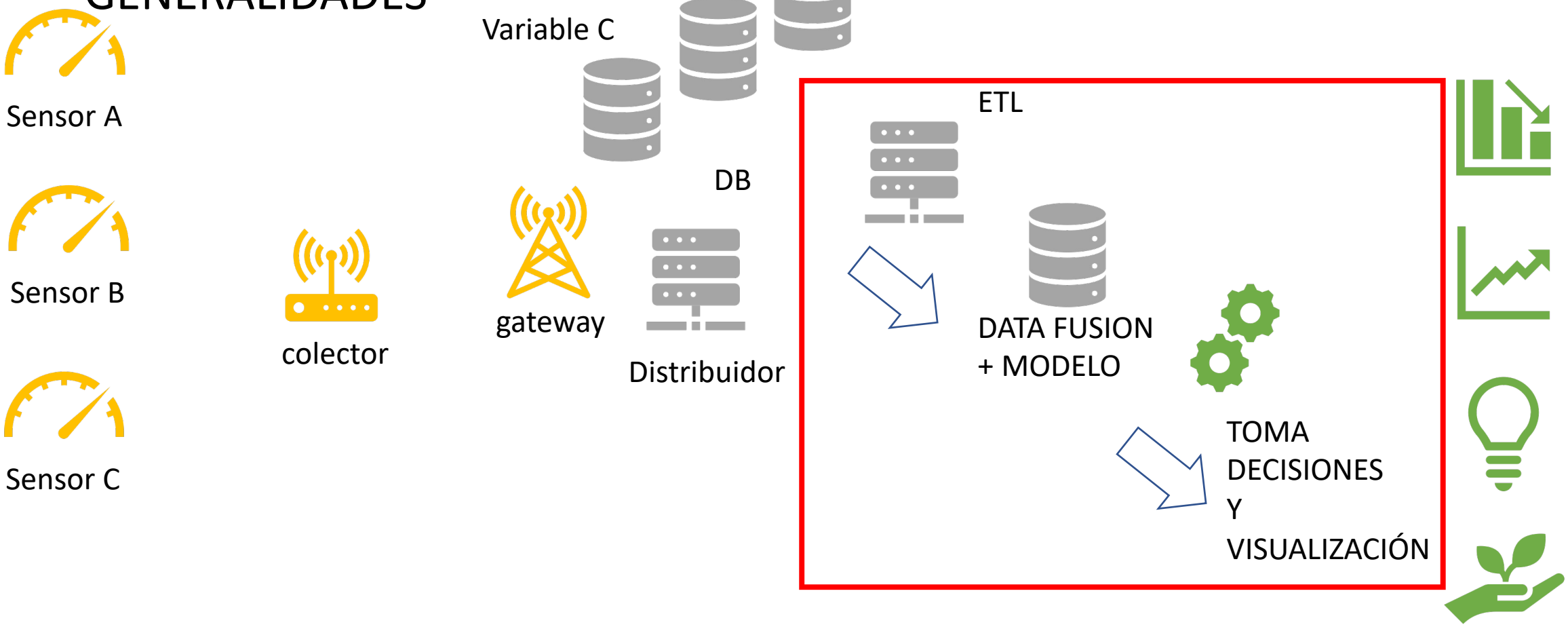
1. Integración de la arquitectura con FiWARE

Sesión 7: 8 horas presenciales

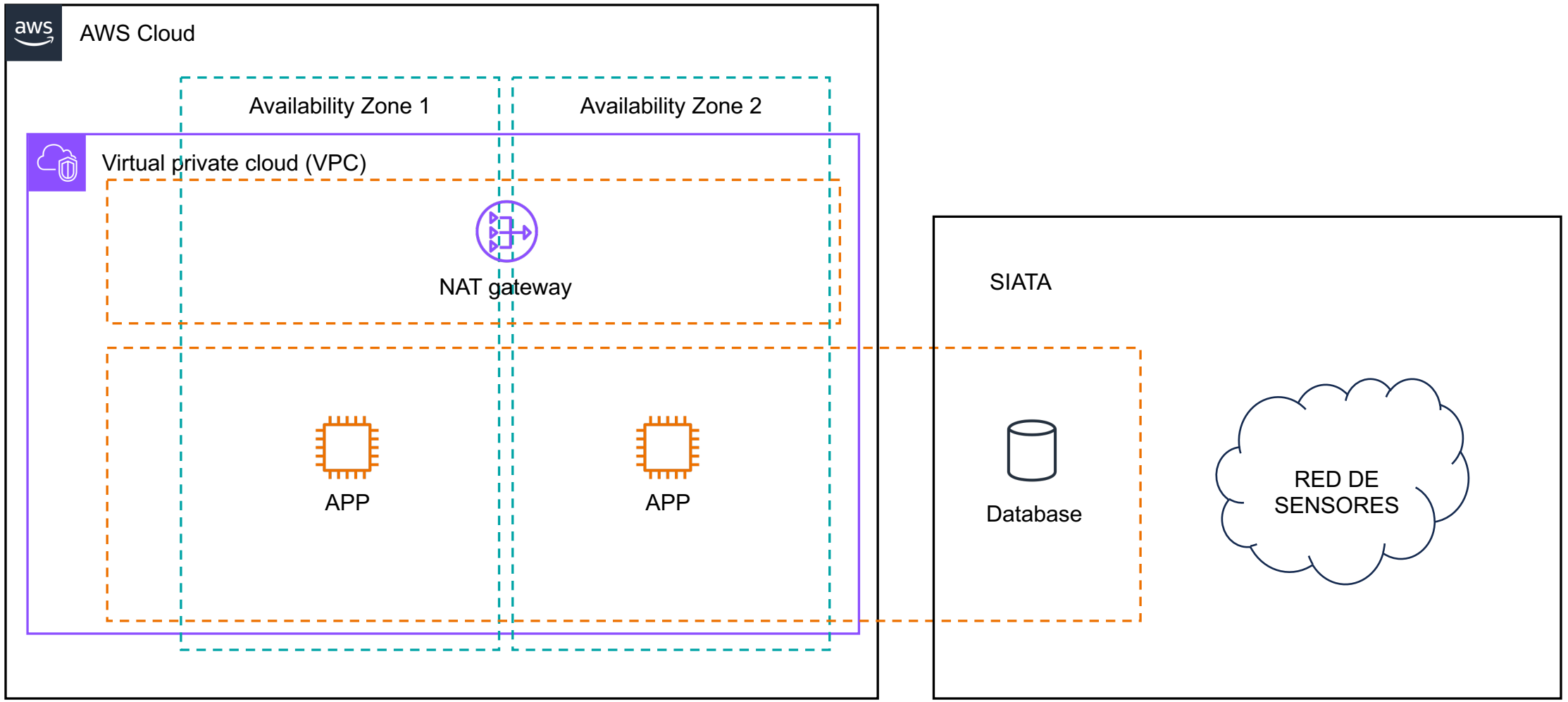
1. Montaje del proyecto presencial
2. Arquitecturas de alta disponibilidad en Nube
3. Ejercicios prácticos de montaje con sensores vía WiFi
4. Dimensionamiento de procesamiento y aspectos financieros de soluciones de Nube

EXTRACCIÓN – TRANSFORMACIÓN Y CARGA

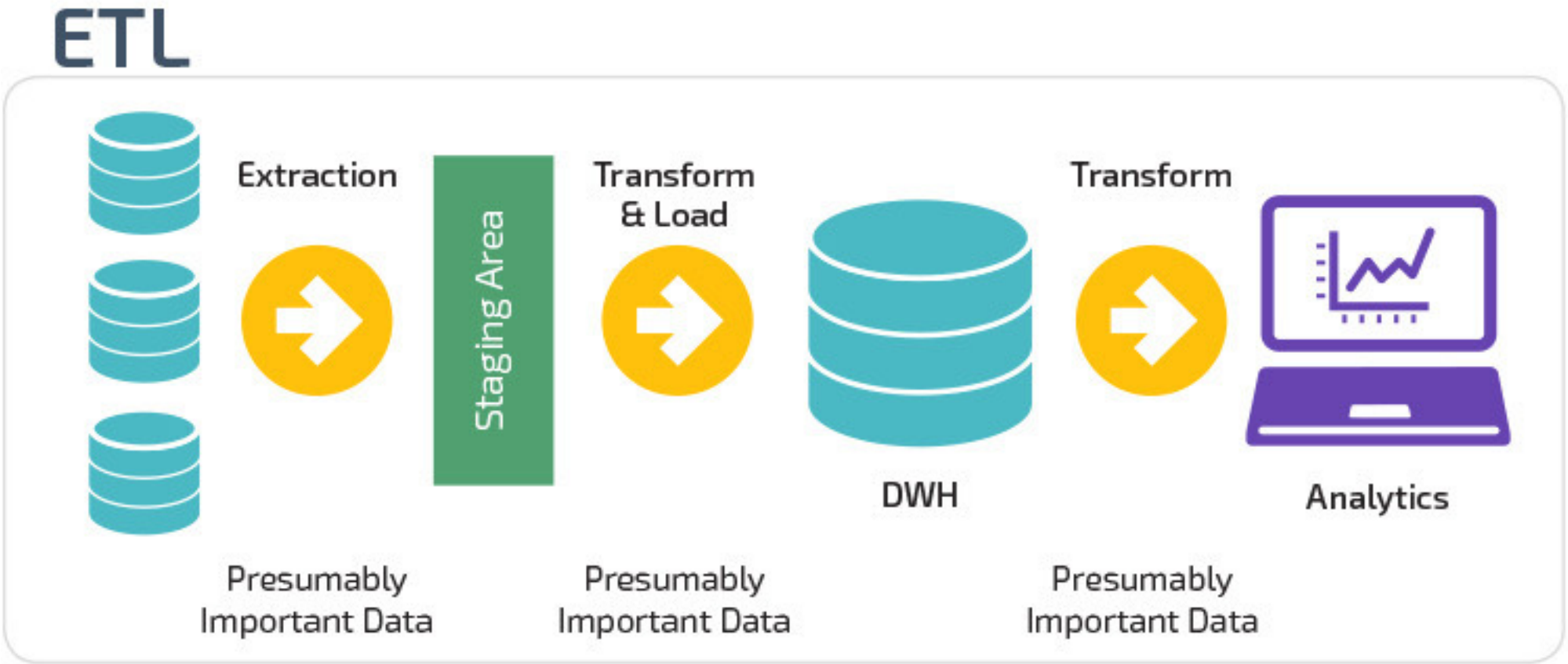
GENERALIDADES



ARQUITECTURA A MONTAR HOY



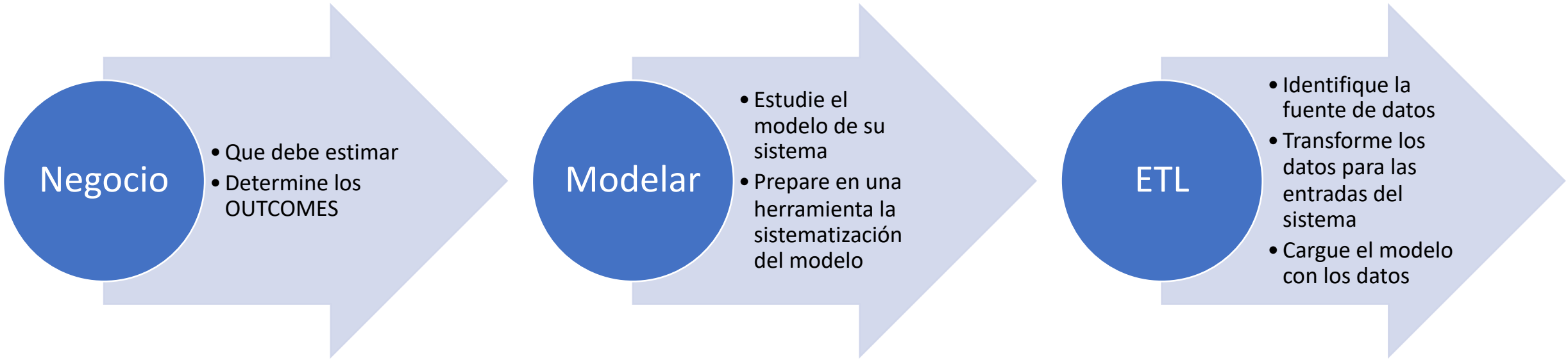
TEORÍA ETL


ETL
Extraction
Staging Area
Transform & Load
Transform
DWH
Analytics
Presumably Important Data
Presumably Important Data
Presumably Important Data

TEORÍA ETL

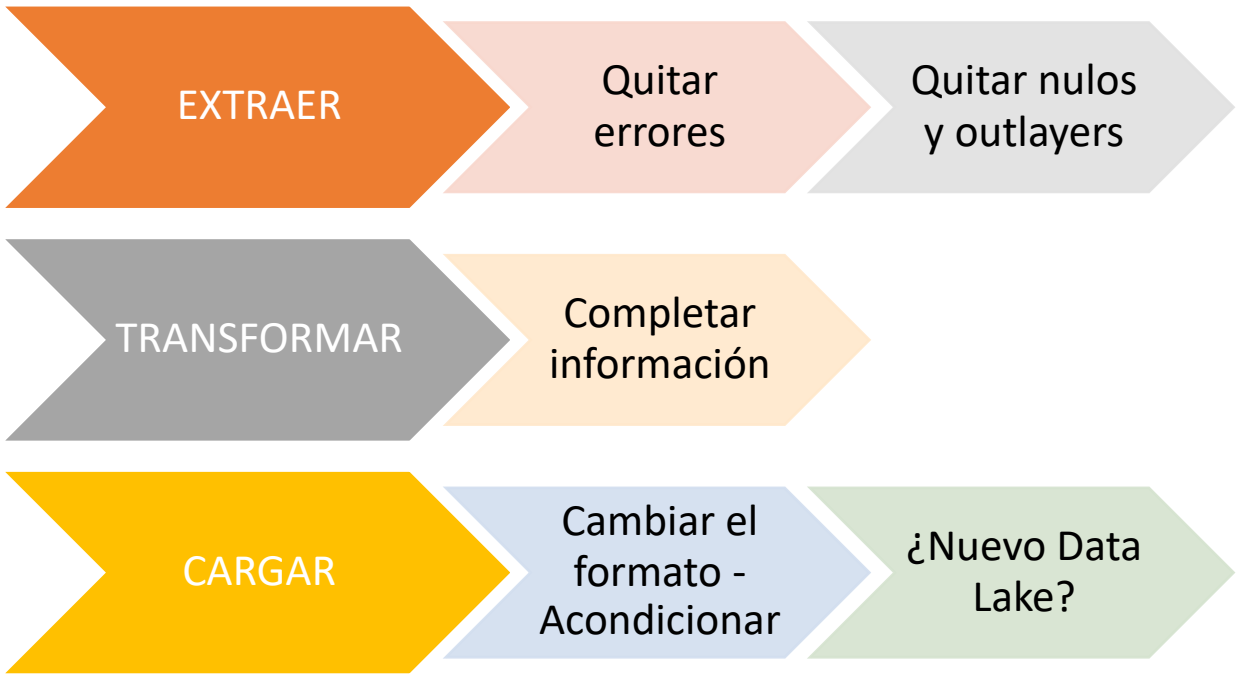
- Debe definir qué es lo importante para su negocio
  - Ejemplo:
    - Estimar la cantidad de producción
    - Identificar amenazas de forma temprana
    - Identificar costos
    - Estimar cantidad de insumos
- Debe modelar el sistema para hallar mecanismos de detección de los elementos a encontrar
- Debe transformar los datos de las fuentes de almacenamiento para aplicarlas al modelo

TEORÍA ETL


Negocio
• Que debe estimar
• Determine los OUTCOMES
Modelar
• Estudie el modelo de su sistema
• Prepare en una herramienta la sistematización del modelo
ETL
• Identifique la fuente de datos
• Transforme los datos para las entradas del sistema
• Cargue el modelo con los datos

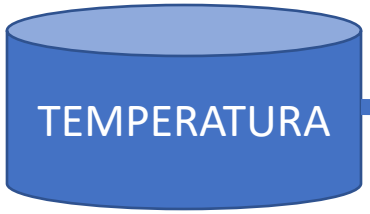
TEORÍA ETL

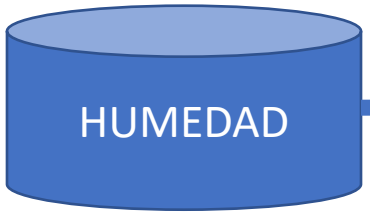
¿QUÉ SE HACE EN LA ETL?



TEORÍA ETL

ID SENSOR	POSICION (X.Y)	TIME STAMP	TEMPERATURA
1	2,4	10AM	24
2	4,3	11AM	26
N	X,Y	12PM	28



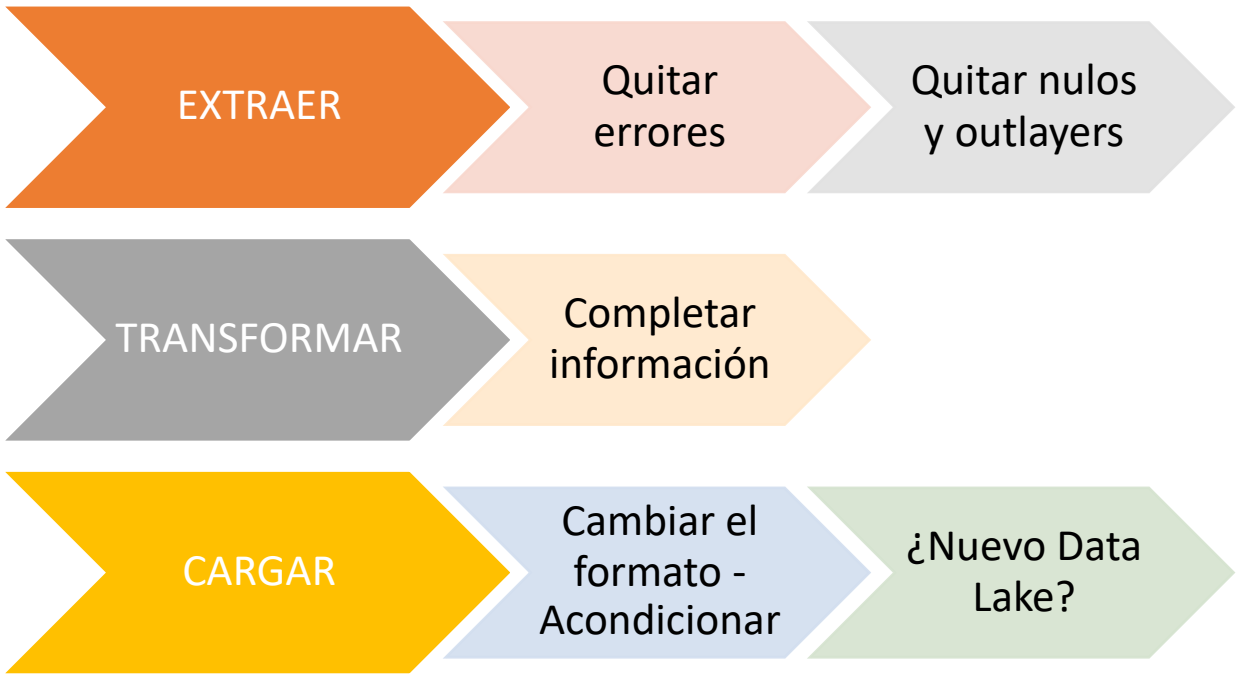


MALLA X.Y MEDIANTE INTERPOLACIÓN

ID SENSOR	POSICION (X.Y)	TIME STAMP	HUMEDAD
1	2,4	10AM	65
2	4,3	11AM	60
N	X,Y	12PM	50

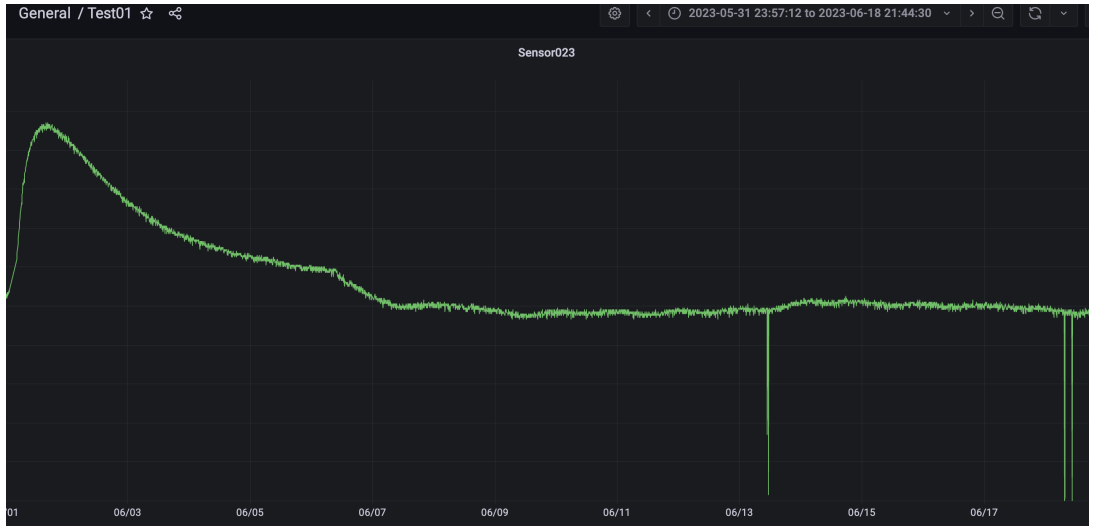
TEORÍA ETL

¿QUÉ SE HACE EN LA ETL?



TEORÍA ETL

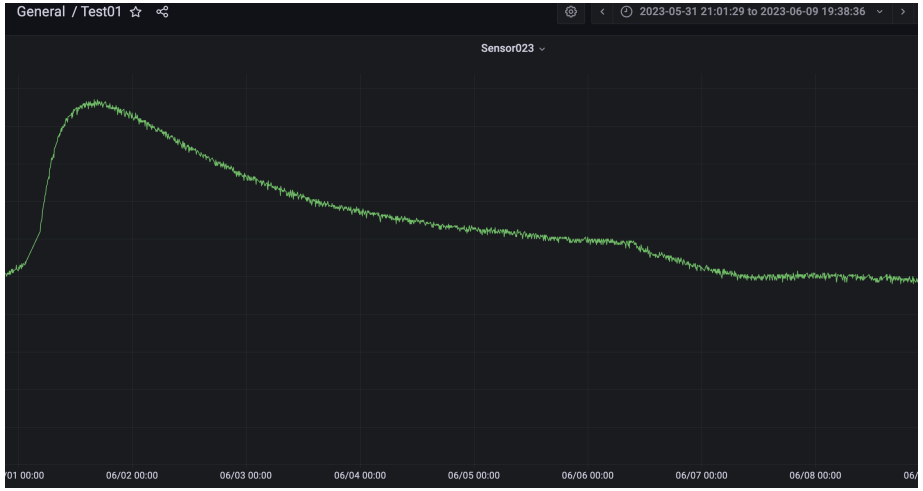
Archivo data.csv




```
import pandas as pd

a = pd.read_csv('data.csv')
a.head()
print(a)

import matplotlib.pyplot as plt

x = a['timestamp'].values
y = a['valor'].values
plt.plot(x,y)
plt.show()

from datetime import datetime

t = pd.to_datetime(a['timestamp'],
unit='ms')
plt.plot(t,y)
plt.show()
```

TEORÍA ETL

Archivo data.csv




```
b = pd.read_csv('data.csv')
indicesborrar = b[ (b['valor'] >= 60) ].index
print(indicesborrar)
b.drop(indicesborrar , inplace=True)
indicesborrar = b[ (b['valor'] < 10) ].index
print(indicesborrar)
b.drop(indicesborrar , inplace=True)
t = pd.to_datetime(b['timestamp'], unit='ms')
y = b['valor'].values
plt.plot(t,y)
plt.figure()
inicio = 2000
fin = 3800
plt.plot(t[inicio:fin],y[inicio:fin])
```

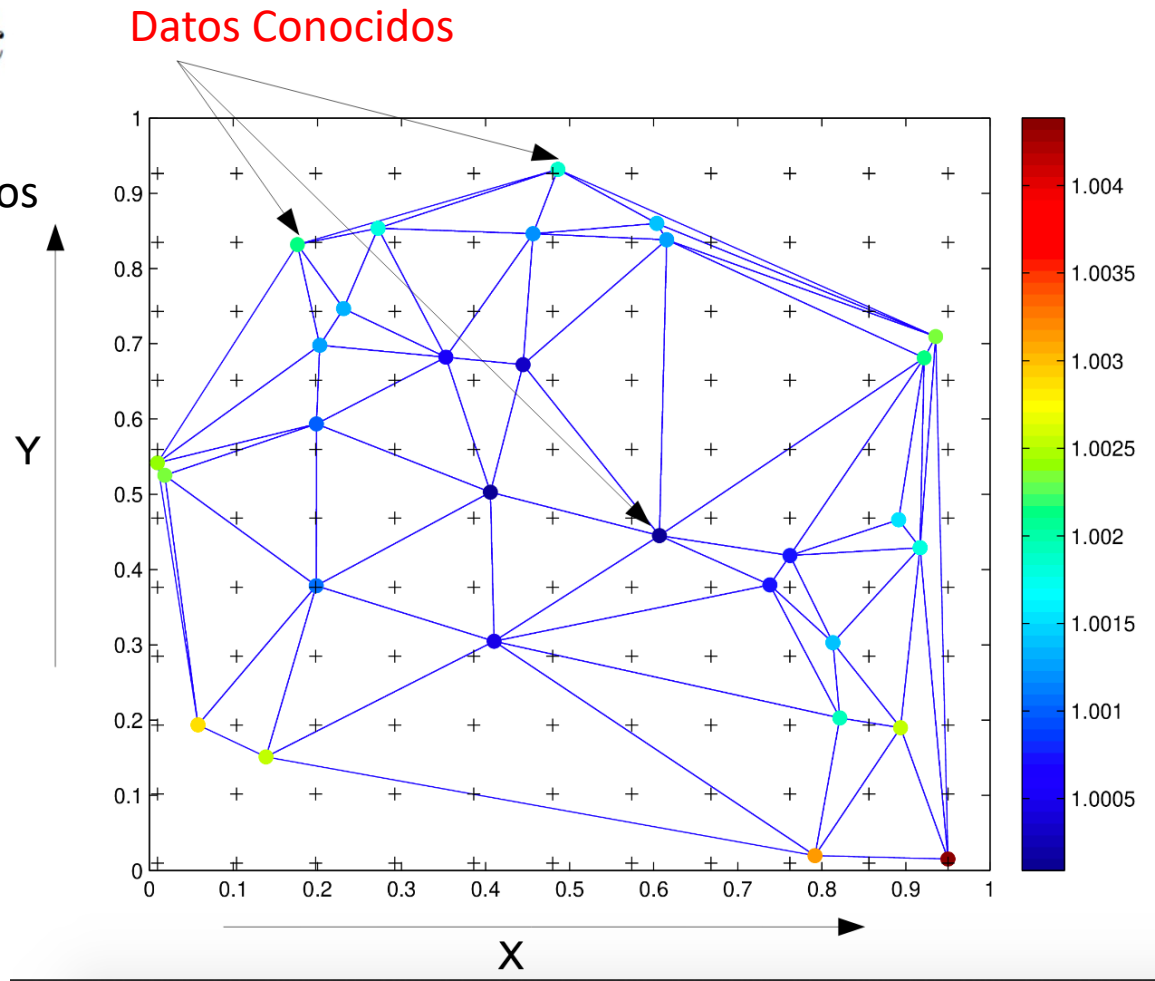
INTERPOLACIÓN

El problema

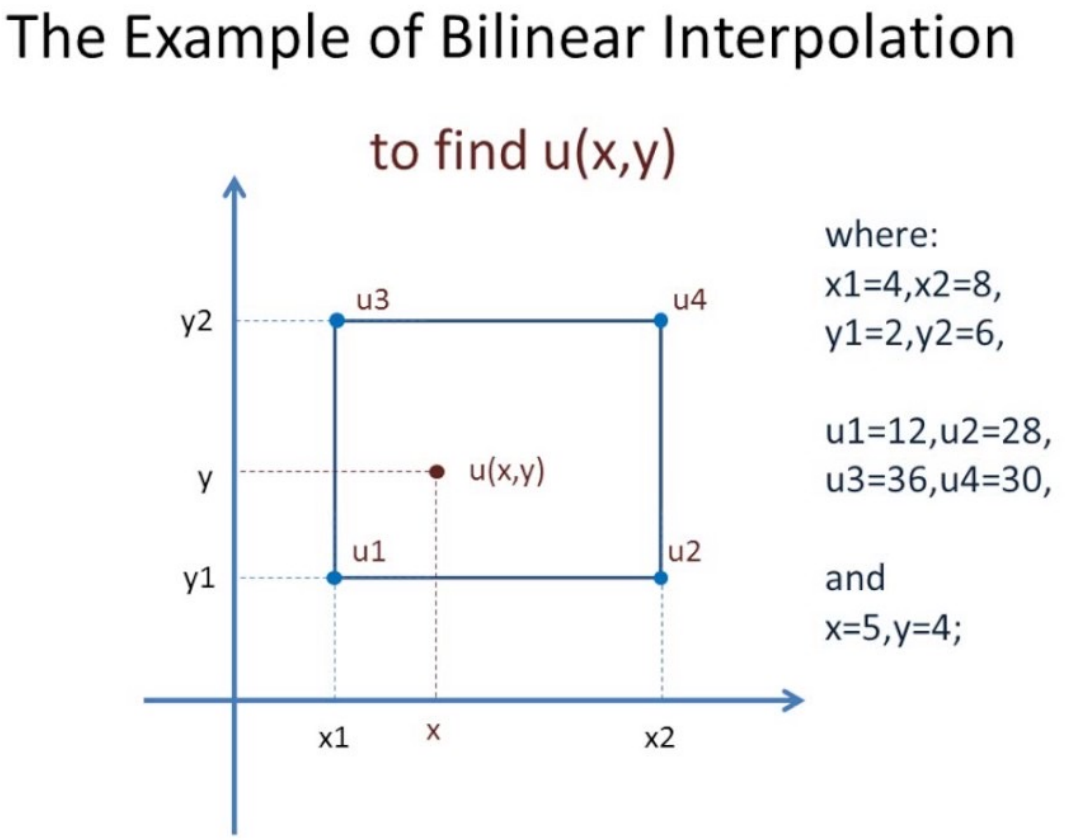
$$\hat{Z}_i = \sum_{k=1}^{k=N} \alpha_{ik} Z_k$$

desconocidos conocidos

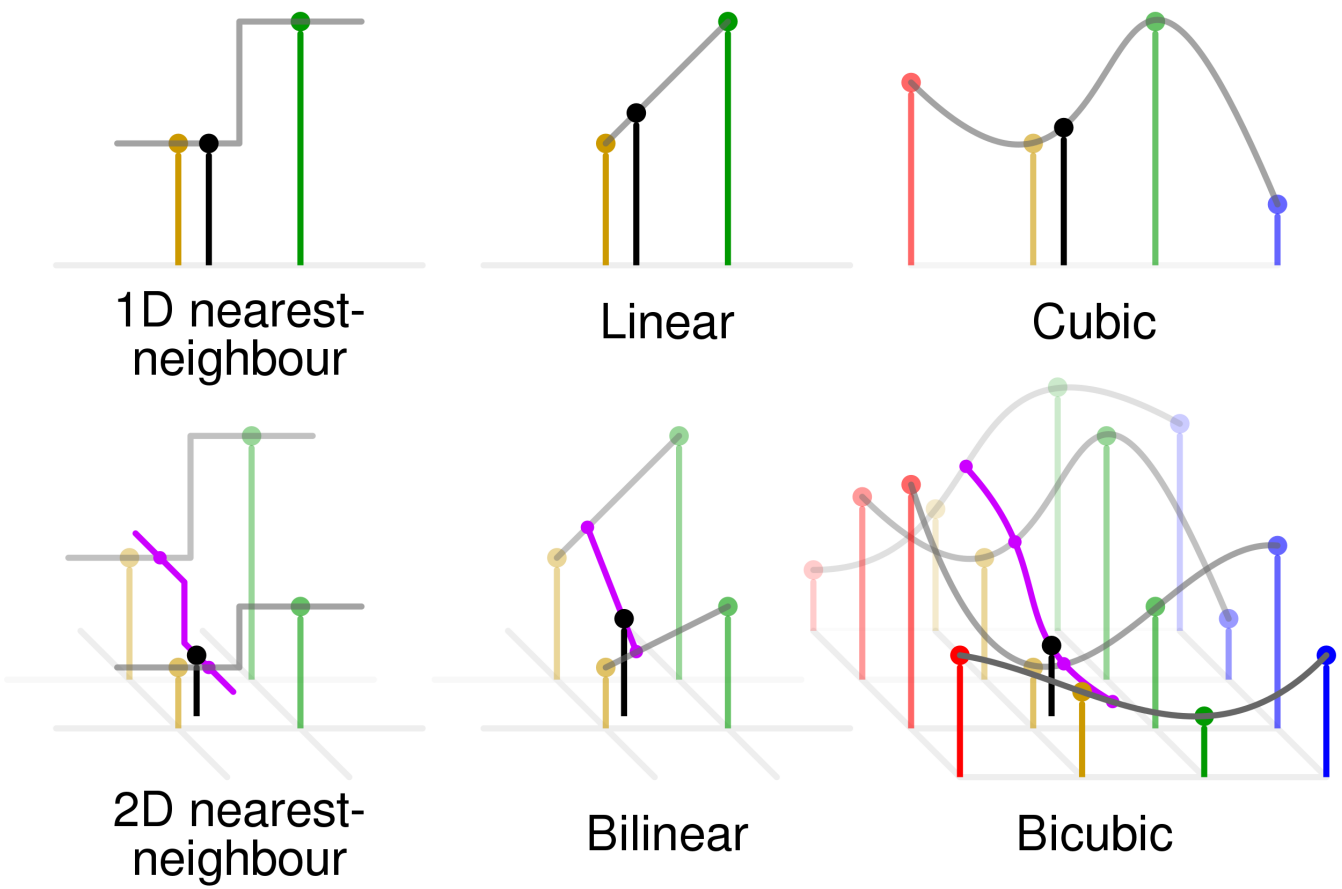
El modelo para estimar el parámetro requiere de un modelo de datos homogeneo (igual tamaño, resolución, matricial, que se le aplica modelos matemáticos para obtener información o decisiones)



INTERPOLACIÓN



INTERPOLACIÓN



USANDO PYTHON COMO HERRAMIENTA

- Usar Python para obtener una malla de datos con unos cuantos valores de unos sensores
  - INTERPOLACION
  - CONTORNOS

- VER LOS EJEMPLOS
  - InterpolandoDatos.py – interpola una malla 2D con unos puntos
  - Solocontonos.py – genera contornos en función de los puntos

USANDO PYTHON COMO HERRAMIENTA

```
#importando la libreria
import numpy as np
#creo mi funcion base (es la que conozco, es mi modelo conocido)
def func(x, y):
 return x*(1-x)*np.cos(4*np.pi*x) * np.sin(4*np.pi*y**2)**2
#creo una malla de 100 x 200
grid_x, grid_y = np.mgrid[0:1:100j, 0:1:200j]
#me invento un grupo de puntos aleatorios
points = np.random.rand(100, 2)
#de estos puntos que serian los sensores, y sus lecturas, voy a interpolar en
#la malla los valores que deseo calcular para mi modelo de deduccion real
values = func(points[:,0], points[:,1])
# ejecuto las interpolaciones, uso tres metodos para comparar su desempeno
from scipy.interpolate import griddata
grid_z0 = griddata(points, values, (grid_x, grid_y), method='nearest')
grid_z1 = griddata(points, values, (grid_x, grid_y), method='linear')
grid_z2 = griddata(points, values, (grid_x, grid_y), method='cubic')
```

librerías

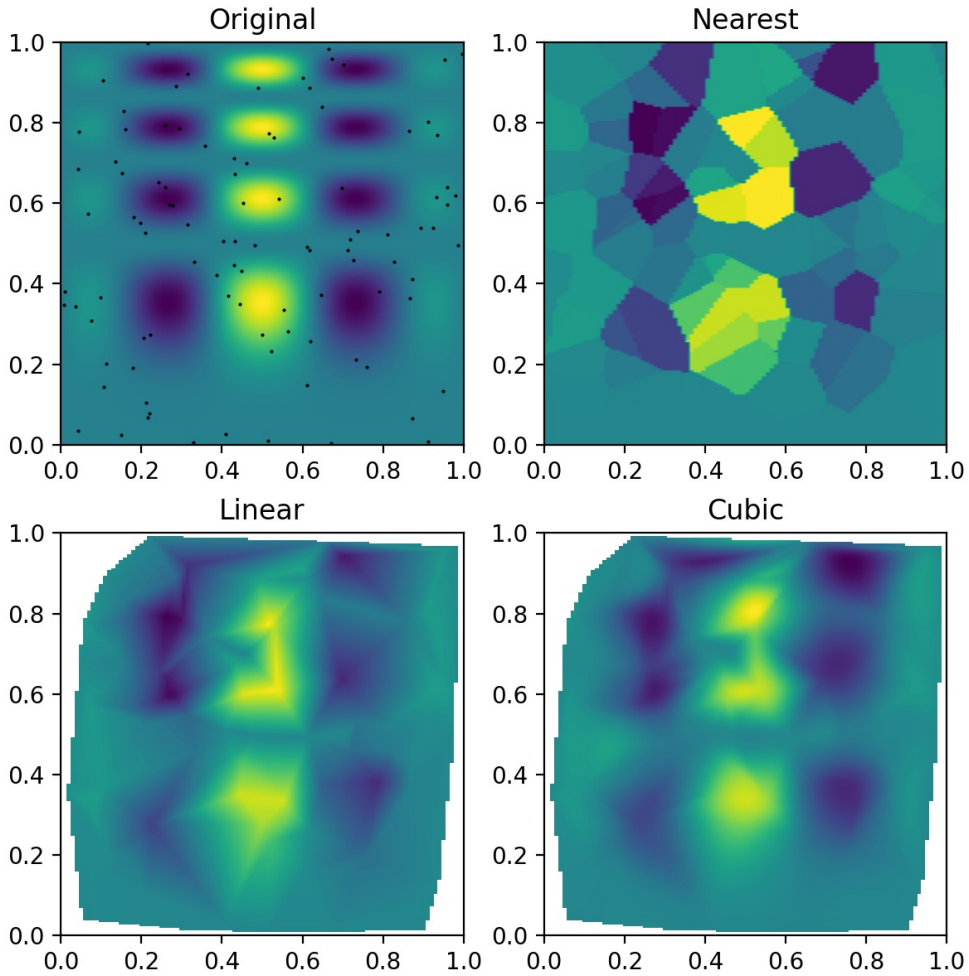
malla

interpolación

```
#grafico los resultados de los tres ejemplos de inteprolacion
import matplotlib.pyplot as plt
plt.subplot(221)
plt.imshow(func(grid_x, grid_y).T, extent=(0,1,0,1), origin='lower')
plt.plot(points[:,0], points[:,1], 'k.', ms=1)
plt.title('Original')
plt.subplot(222)
plt.imshow(grid_z0.T, extent=(0,1,0,1), origin='lower')
plt.title('Nearest')
plt.subplot(223)
plt.imshow(grid_z1.T, extent=(0,1,0,1), origin='lower')
plt.title('Linear')
plt.subplot(224)
plt.imshow(grid_z2.T, extent=(0,1,0,1), origin='lower')
plt.title('Cubic')
plt.gcf().set_size_inches(6, 6)
plt.show()
```

Comparando graficas

USANDO PYTHON COMO HERRAMIENTA



USANDO PYTHON COMO HERRAMIENTA

```
# Cargando la libreria
import matplotlib.pyplot as plt
import matplotlib.tri as tri
import numpy as np
np.random.seed(19680801)
npts = 200
ngridx = 100
ngridy = 200
x = np.random.uniform(-2, 2, npts)
y = np.random.uniform(-2, 2, npts)
z = x * np.exp(-x**2 - y**2)
fig, (ax1, ax2) = plt.subplots(nrows=2)
# Interpolando una malla
# se crea la malla
xi = np.linspace(-2.1, 2.1, ngridx)
yi = np.linspace(-2.1, 2.1, ngridy)
# se hace una interpolacion lineal con los datos x,y
# en una grilla definida por espacios xi, yi
triang = tri.Triangulation(x, y)
interpolator = tri.LinearTriInterpolator(triang, z)
Xi, Yi = np.meshgrid(xi, yi)
zi = interpolator(Xi, Yi)
```

librerías

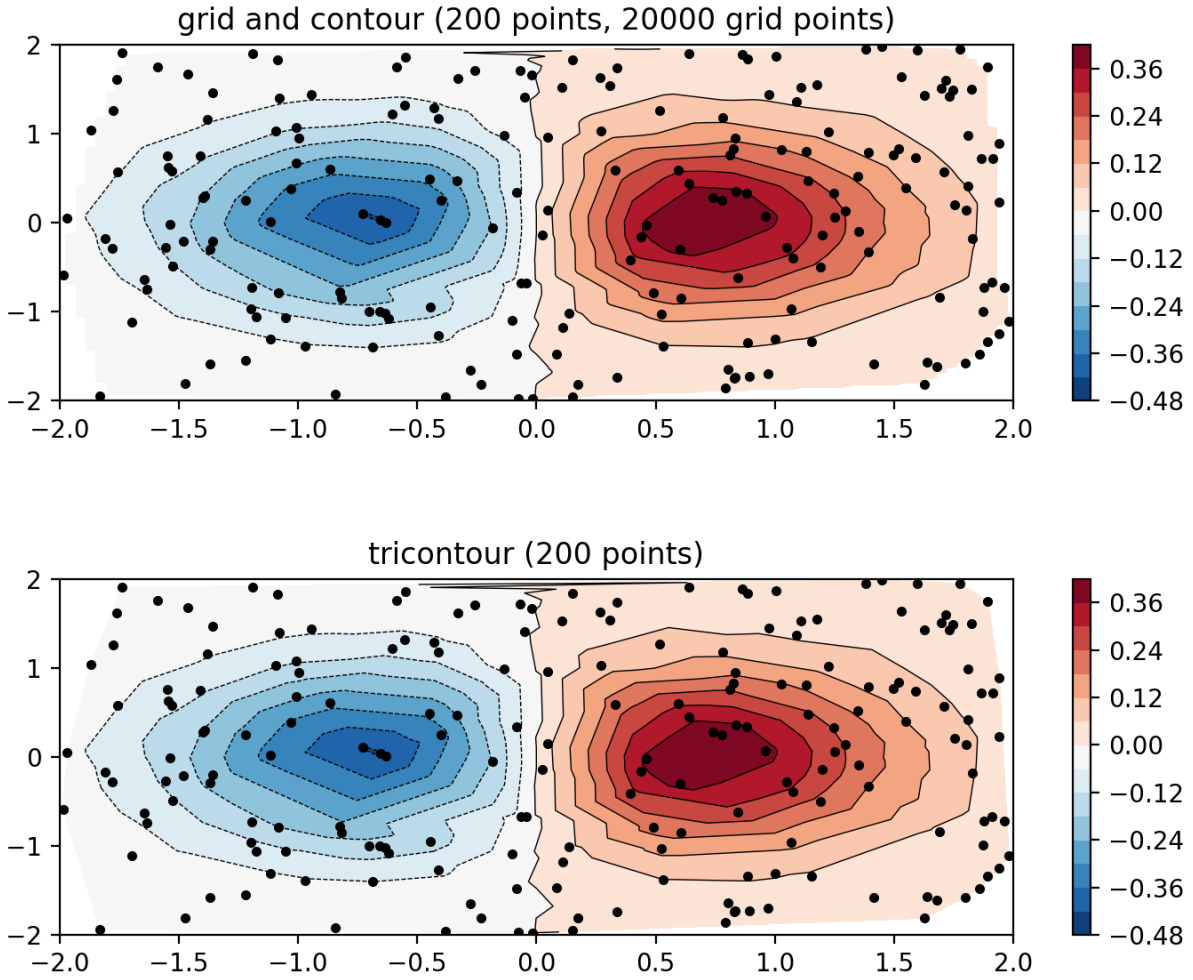
malla

interpolación

```
ax1.contour(xi, yi, zi, levels=14, linewidths=0.5, colors='k')
cntr1 = ax1.contourf(xi, yi, zi, levels=14, cmap="RdBu_r")

fig.colorbar(cntr1, ax=ax1)
ax1.plot(x, y, 'ko', ms=3)
ax1.set(xlim=(-2, 2), ylim=(-2, 2))
ax1.set_title('grid and contour (%d points, %d grid points)' %
              (npts, ngridx * ngridy))

# aplicamos la tecnica para generar los contornos
ax2.tricontour(x, y, z, levels=14, linewidths=0.5, colors='k')
cntr2 = ax2.tricontourf(x, y, z, levels=14, cmap="RdBu_r")

fig.colorbar(cntr2, ax=ax2)
ax2.plot(x, y, 'ko', ms=3)
ax2.set(xlim=(-2, 2), ylim=(-2, 2))
ax2.set_title('tricontour (%d points)' % npts)

plt.subplots_adjust(hspace=0.5)
plt.show()
```

Contornos

USANDO PYTHON COMO HERRAMIENTA



USANDO PYTHON COMO HERRAMIENTA

EJERCICIO

Tome la base de datos de las particulas 2.5m de la contaminación de la ciudad de Medellín, y construya una grilla que muestre los contornos y los niveles de intensidad de la ultima medición disponible en una resolución de 100 metros de distancia.

EJERCICIOS

1. CREE UN DASH APP EN PYTHON Y MUESTRE LA CONTAMINACIÓN DEL AIRE EN MEDELLIN EN UNA APP WEB
2. AUTOMATICE LA OPERACIÓN EN UN DOCKER FILE
3. LANCE LA INSTANCIA COMO SERVIDOR INDIVIDUAL
4. LANCE VARIAS INSTANCIAS COMO SERVIDORES EN EL MISMO PUERTO – USE DOCKER
    1. ¿CÓMO SE HARIA CON GITHUB?
5. CONECTE UN BALANCEADOR DE CARGA