# PERSISTENCIA DE DATOS CON Python

# 3

@SENAComunica

www.sena.edu.co

GC-F-004v.08

# CAPA DE PERSISTENCIA
# Base de Datos

jgalindos@sena.edu.co 2023

# SOFTWARE RECOMENDADO

- MARIADB SERVER
- POSTGRES SERVER
- MONGO SERVER
- SQLITE SERVER
- REPOSITORIO GITHUB

JOSE FERNANDO GALINDO SUAREZ

# AGENDA DE TEMAS

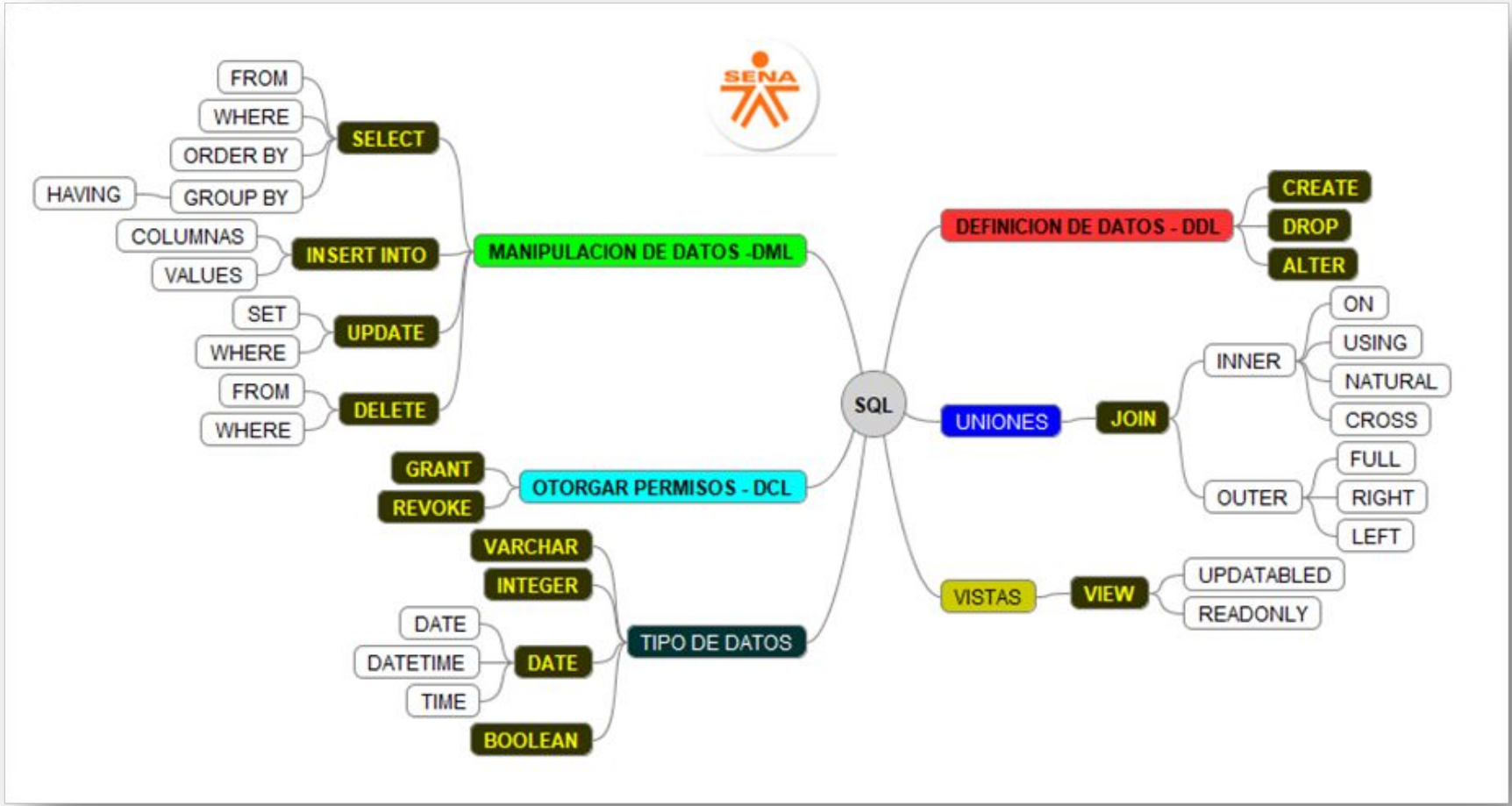
- LENGUAJE DE CONSULTA ESTRUCTURADA
- CONECTANDO CON MYSQL
- MIGRANDO DATOS DESDE EXCEL A MYSQL CON PYTHON
- CONECTNADO MONGODB CON PYTHON
- CONECTNADO POSTGRESQL CON PYTHON
- OBJECT RELATIONAL MAPPING (ORM)
- DOCKER
- MICROSERVICIOS CON PYTHON
- CONECTANDO CON API

VERSION PDF

JOSE FERNANDO GALINDO SUAREZ

# LENGUAJE DE CONSULTA ESTRUCTURADA

# Lenguaje de Consulta estructurada



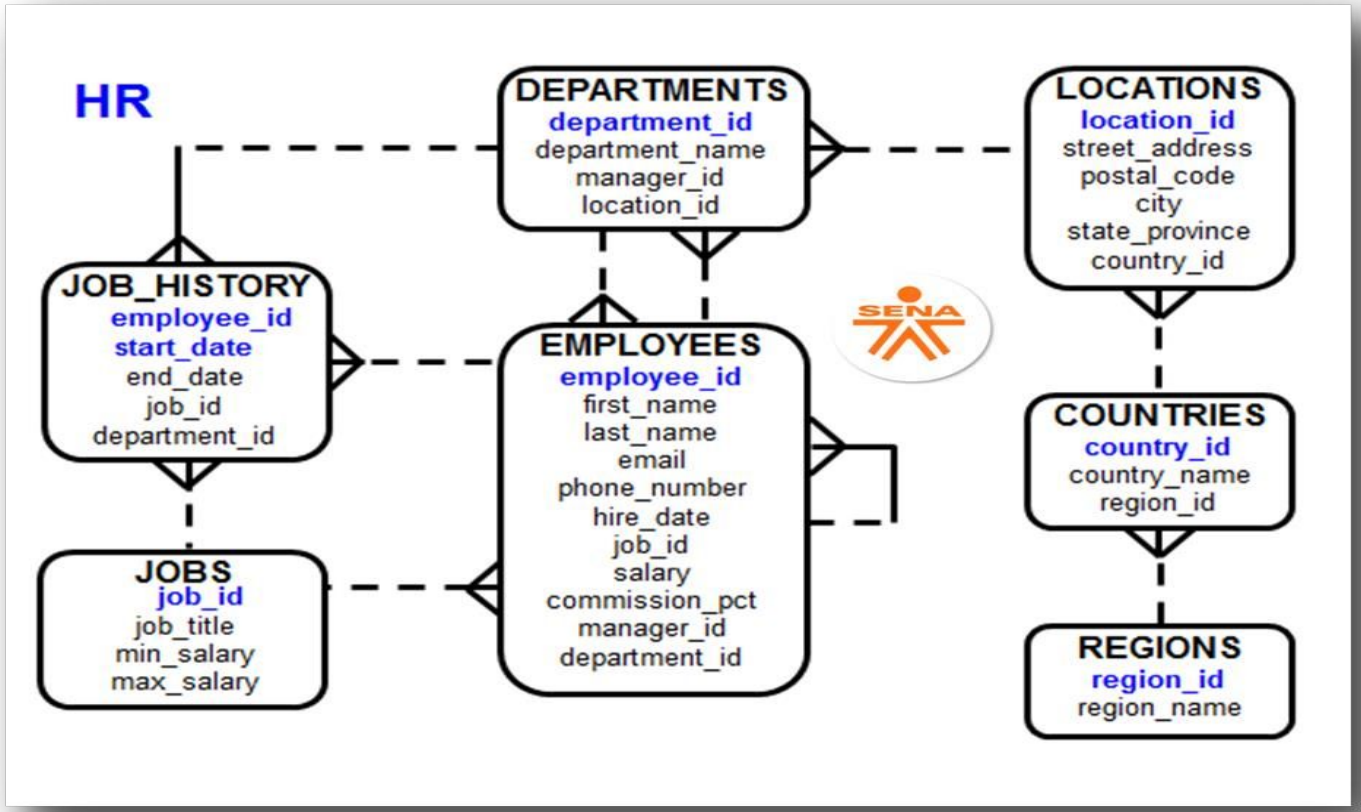

JOSE FERNANDO GALINDO SUAREZ

# Conectando al servidor MySQL

```
# mysql -u root -p
Enter password: ****
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 12
Server version: 10.1.37-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

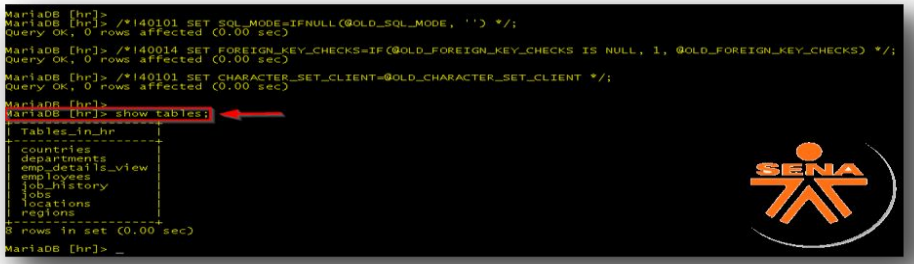
JOSE FERNANDO GALINDO SUAREZ

# Crear el esquema de base de datos




- **Descargar el script de creación de la base de datos HR Copiar el script al block de notas.**

- **Pegar desde el portapapeles a la consola Mysql**

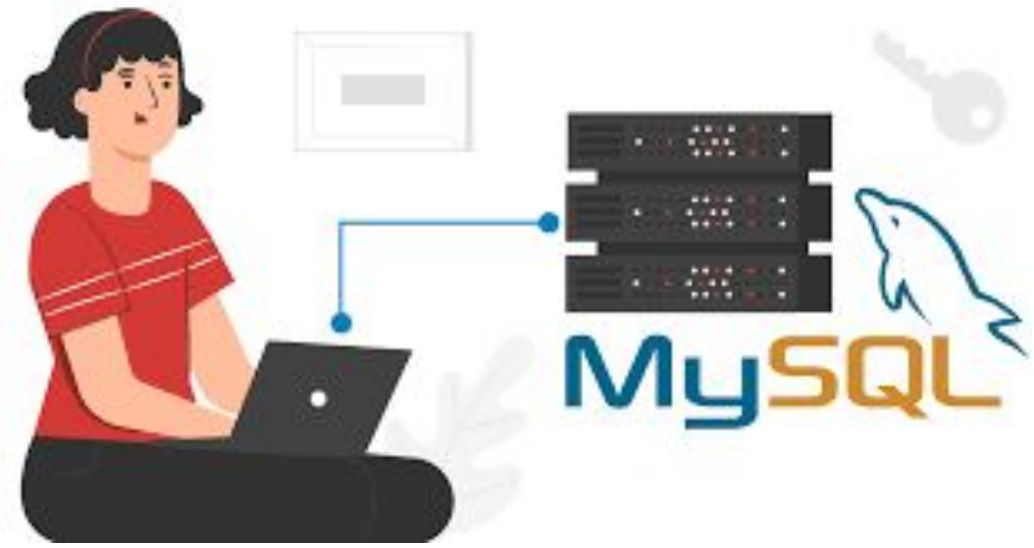



**HR DATASET**

# CONECTANDO MYSQL CON PYTHON

# Conectando MySQL Python

```
pip install mysql-connector-python

Importar las bibliotecas necesarias
import mysql.connector
import pandas as pd

Realizar la conexión hacia el servidor mysql
conn=mysql.connector.connect(  host="localhost",
user="root",
passwd="",  database="hr",  port=3306)
```




JOSE FERNANDO GALINDO SUAREZ

# Conectando MySQL Python

**Crear un cursor para interactuar con MySQL**

```
cursor=conn.cursor()
```

**Obtener todas las filas de la tabla employees**

```
sq="select * from employees"
cursor.execute(sq)
resultado=pd.DataFrame(cursor.fetchall())
print(resultado)
```

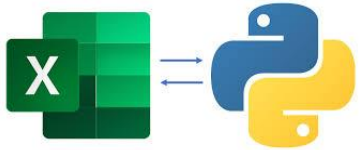
JOSE FERNANDO CALINDO SUAREZ

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

```
#Instalar los siguientes drivers desde la línea de comando y descargar el archivo Neptuno.xlsx:
pip install mysql-connector-python
#pip install xlrd
```

```
#Importar las bibliotecas necesarias
import mysql.connector
import xlrd
import pandas as pd
```

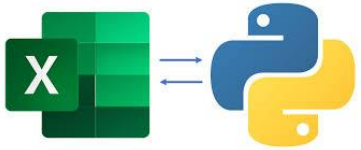


```
#Realizar la conexión hacia el servidor mysql
conn=mysql.connector.connect(host="localhost",
user="root",
passwd="",
database="neptuno",
port=3306
)
```

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

```
import mysql.connector
import xlrd
import pandas as pd

con=mysql.connector.connect(host="localhost",
user="root",
passwd="",
database="prueba",
port=3306
)

cursor=con.cursor()
sq="""CREATE TABLE IF NOT EXISTS CATEGORIA(
        IDCATEGORIA INTEGER PRIMARY KEY, NOMBRECATEGORIA TEXT, DESCRIPCION TEXT)"""

cursor.execute(sq)
con.commit()
```



JOSE FERNANDO GALINDO SUAREZ

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

## INSERTAR REGISTROS EN LA TABLA CATEGORIAS

**Cargar la hoja “2,-Categorias” del archivo Excel “Neptuno.xls”**

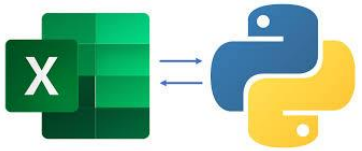
```
x=pd.read_excel(“Neptuno.xlsx”,shee_name=“2,-Categorias”)
```

**Se cargan los nombres de las columnas**

```
k=[]
for i in x:
 k.append(i)
z0=x[k[0]]
z1=x[k[1]]
Z2=x[k[2]]
```

**Cantidad de filas**

```
j=len(x)
```



JOSE FERNANDO GALINDO SUAREZ

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

## INSERTAR REGISTROS EN LA TABLA CATEGORIAS

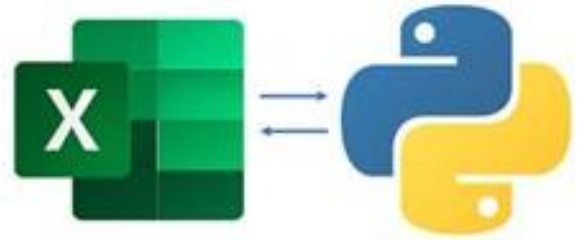
**Recorrer las filas para construir la instrucción insert y se ejecuta para  crear el registro”**

```
 j=len(x)
i=0
while i<j:
   sql="insert into categoria(idcategoria,NombreCategoria,Descripcion)
values({0},'{1}','{2}')".format(i,z0[i],z1[i])
   # sqll=(str(i),z0[i],z1[i])
   print(sql )
   i=i+1
   cursor.execute(sql)
   con.commit()
```

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA MIGRANDO DESDE EXCEL CON PYTHON

## INSERTAR REGISTROS EN LA TABLA CATEGORIAS



```
x=pd.read_excel("Neptuno.xlsx",sheet_name="2,-Categorias")
k=[]
for i in x:
    k.append(i)
k.append("idcategoria")
for i in range(len(x)):
    kk=[]
    colu=str(k)
    kk.append(x.loc[i][0])
    kk.append(x.loc[i][1])
    kk.append(i+1)
    kk=str(kk).replace("[","").replace("]","")
    colu=colu.replace("[","").replace("'","").replace("]","")
    sql="insert into categoria("+colu+") values("+kk+")"
    cursor.execute(sql)
    con.commit()
```

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA CONECTANDO MONGODB CON PYTHON

python mongo

# CONECTANDO MONGODB CON PYTHON

**Conectando Python con MongoDB**

**Conectar al servidor MongoDB local**

```
import pymongo
from pymongo import MongoClient

CONNECTION_STRING = "mongodb://localhost:27017/"

# Create a connection using MongoClient. You can import MongoClient or use pymongo.MongoClient
client = MongoClient(CONNECTION_STRING)
```

**Crear la base de datos llamada “titulada” y la colección “aprendices”**

```
dbname = client["titulada"]
coleccion=dbname["aprendices"]
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MONGODB CON PYTHON

**Conectando Python co MongoDB**

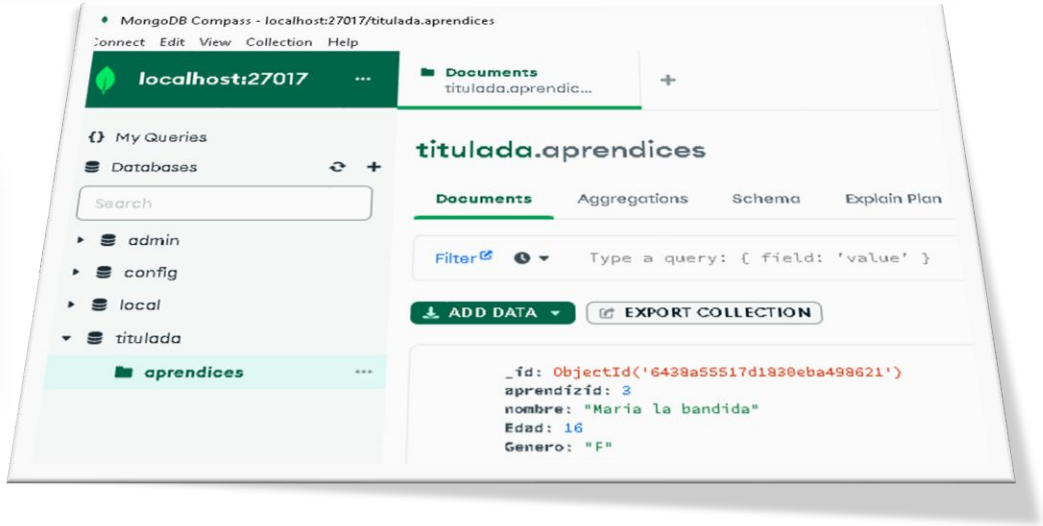
**Insertar un documento**

```
coleccion.insert_one({"aprendizid":3,"nombre":"Maria la bandida","Edad":16,"Genero":"F"})
```

**Observar en MongoDB Compas**

**Inserta más de un documento**

```
coleccion.insert_many([{"aprendizid":4,"nombre":"Rosa Melo Rico","Edad":36,"Genero":"F"},
{"aprendizid":5,"nombre":"Simon Tolomeo","Edad":55,"Genero":"M"}])
```




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MONGODB CON PYTHON

**Conectando Python con MongoDB**

**Mostrar todos los documentos de la colección**

```
a=coleccion.find()
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

**Mostrar un registro en especial**

```
a=coleccion.find({"aprendizid":3})
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

**Mostrar todos los documentos ordenados por la edad de forma ascendente**

```
a=coleccion.find().sort("Edad",1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

**Mostrar todos los documentos ordenados por la edad de forma descendente**

```
a=coleccion.find().sort("Edad",-1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

**Cambiar el nombre del aprendiz 3 por "García Rovira"**

```
coleccion.update_one({"aprendizid":3},
                     {"$set":{"nombre":"García Rovira"}})
```

JOSE FERNANDO CALUNDO SUAREZ

# CONECTANDO MONGODB CON PYTHON

**Conectando Python con MongoDB**

**Cambiar el género por “M” a los que el nombre comience por “Ro”**

```
myquery = { "nombre": { "$regex": "^Ro" } }
newvalues = { "$set": { "Genero": "M" } }

coleccion.update_many(myquery,newvalues)
```

**Listar solamente el mayor de los aprendices**

```
a=coleccion.find().sort("Edad",-1).limit(1)
for x in a:
    print(x["aprendizid"],x["nombre"],x["Genero"],x["Edad"])
```

**Borrar el aprendiz cuyo id es igual a 5**

```
coleccion.delete_one({"aprendizid":5})
```

**Borrar la colección aprendiz**

```
coleccion.drop()
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MONGODB CON PYTHON

**Importar las librería necesarias**

```
import pymongo
from pymongo import MongoClient
import pandas as pd
```

**Realizar la conexión al servidor MongoDB**

```
cadena="mongodb://localhost:27017"
cliente=MongoClient(cadena)
```

**Cargar la hoja "2,-Categorias" del archivo Excel "Neptuno.xls"**

```
x=pd.read_excel("c:/Borrar/Neptuno.xls",sheet_name="2,-Categorias")
```

**Cargar los nombres de las columnas**

```
k=[]
for i in x:
    k.append(i)

z0=x[k[0]]
z1=x[k[1]]
z2=x[k[2]]
```

**Cantidad de filas**

```
j=len(x)
```

**Crear la base de datos "Neptuno.xls"**

```
bd=cliente["Neptuno"]
```

**Se crear la colección.**

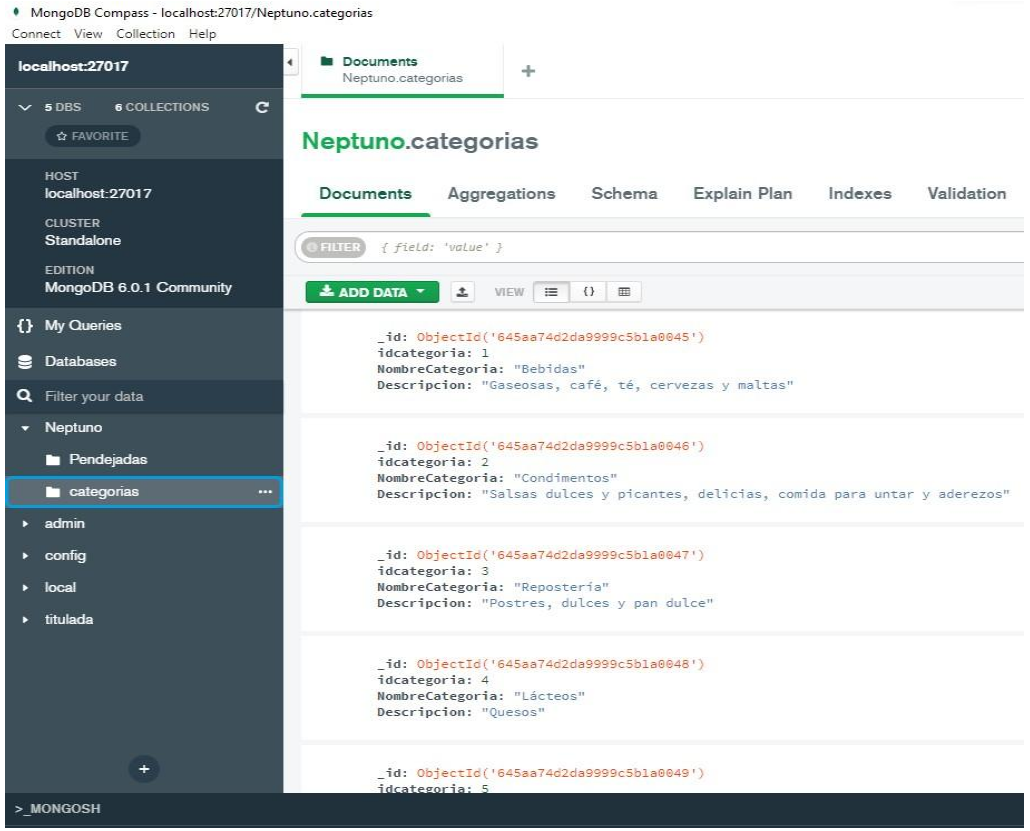
```
coleccion=bd["categorias"]
```

**Insertar todas filas de la hoja del libro Excel**

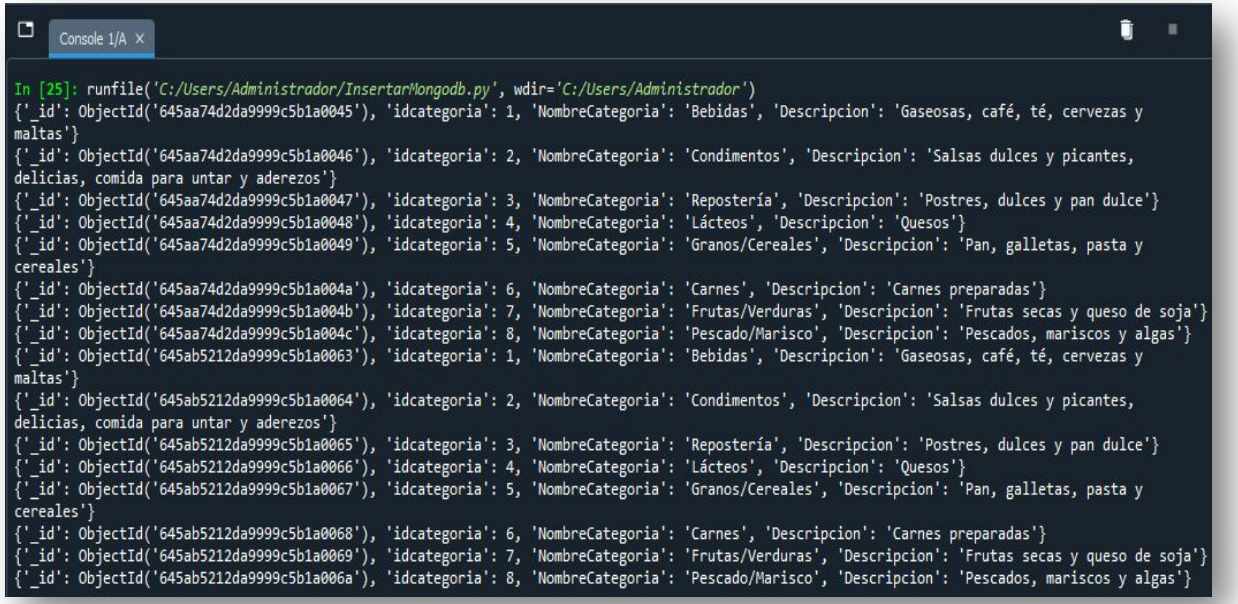
```
i=0
while i<j:
    coleccion.insert_one({"idcategoria":int(z0[i]),"NombreCategoria":z1[i],"Descripcion":z2[i]})
    i=i+1
```

# CONECTANDO MONGODB CON PYTHON

**Verificar en MongoDb si se creo la base de datos "Neptuno", la colección "categorías"**




**Mostrar todos los documentos de la colección "categorías"**




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MONGODB CON PYTHON

python mongo

**Mostrar el "NombreCategoria"**

```
Docu=coleccion.find()
for Muestre in Docu:
    print(Muestre['NombreCategoria'])
```

```
In [29]: runfile('C:/Users/Administrador/InsertarMongodb.py', wdir='C:/Users/Administrador')
Bebidas
Condimentos
Repostería
Lácteos
Granos/Cereales
Carnes
Frutas/Verduras
Pescado/Marisco
```

mongo DB

JOSE FERNANDO GALINDO SUAREZ

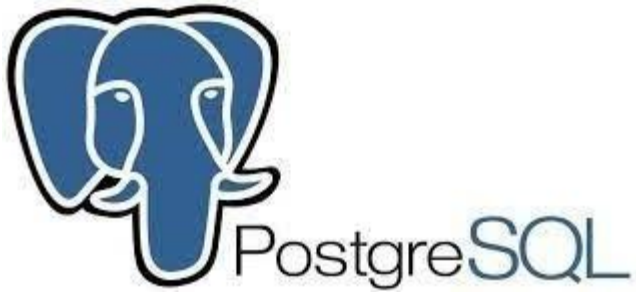
# CONECTANDO POSTGRESQL CON PYTHON

# PRACTICA CON POSTGRESQL

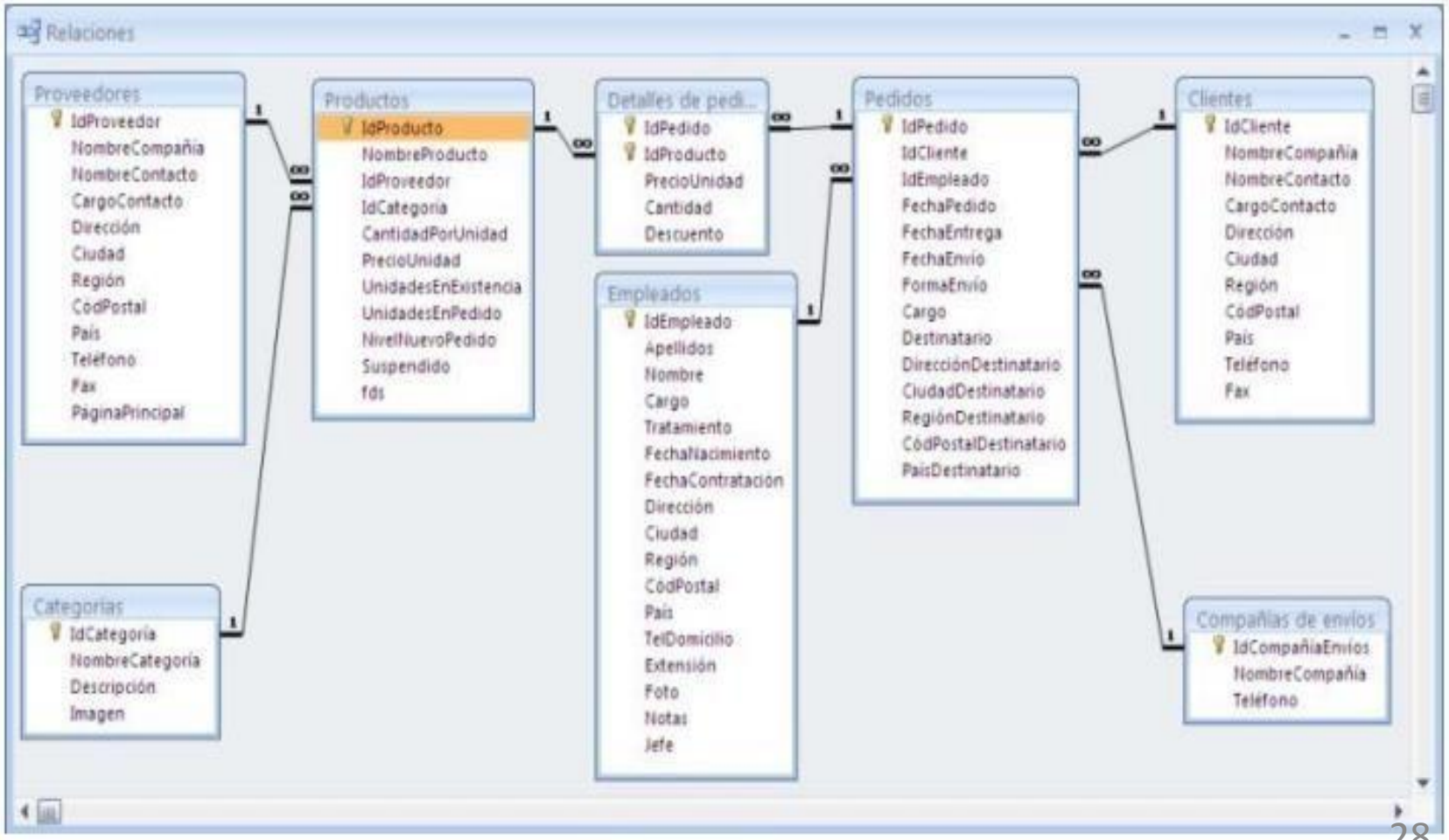
Descargar PostgreSQL y Neptuno e instálelo para realizar la practica con el modelo relacional propuesto.
Importar las librerías necesarias para realizar la práctica

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON

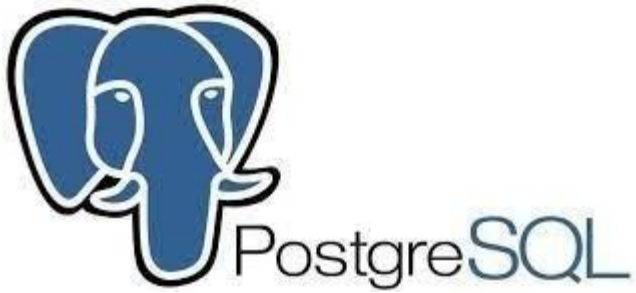



# NEPTUNO




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON




**Realizar la conexión al servidor Postgres**

```
import psycopg2
import pandas as pd
```

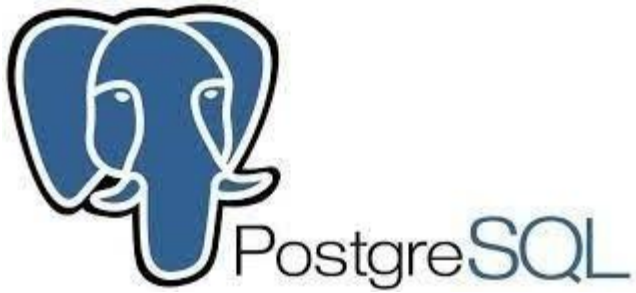
```
conn=psycopg2.connect(
host="localhost",
    user="postgres",
    password="",
    database="neptuno"
)
```

**Crear un cursor**

```
micursor=conn.cursor()
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON




**Crear la table CATEGORIA en Postgres**

```
sq="CREATE TABLE IF NOT EXISTS CATEGORIA(IDCATEGORIA SERIAL
PRIMARY
KEY,NOMBRECATEGORIA TEXT, DESCRIPCION TEXT)"

micursor.execute(sq)
conn.commit()
```

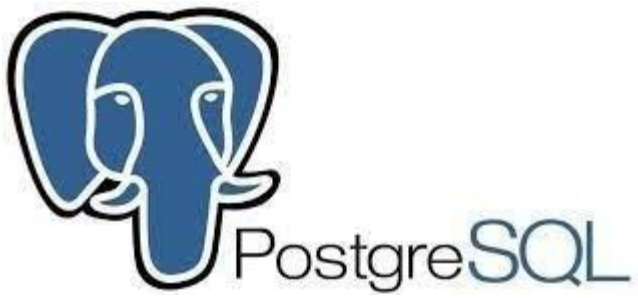
**Insertar un registro en la tabla CATEGORIA en Postgres**

```
sq="insert into categoria(nombrecategoria,descripcion)
values('Panaderia','Pan quemado')”

micursor.execute(sq)
conn.commit()
```

```
sq="update categoria set descripcion='Pan Horneado' where
idcategoria=1“

micursor.execute(sq)
conn.commit()
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON




**Borrar un registro de la tabla CATEGORIA en Postgres**

```
sq="delete from categoria where idcategoria=1"
```

```
micursor.execute(sq)
conn.commit()
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON

**Extraer los datos del libro “Neptuno.xls” de la hoja 5**

```
x=pd.read_excel('c:/Borrar/Neptuno.xlsx',sheet_name
=
5)
```

| | IdCategoría | NombreCategoría | Descripción |
|---|---|---|---|
| 0 | 5 | Granos/Cereales | Pan, galletas, pasta y cereales |
| 1 | 7 | Frutas/Verduras | Frutas secas y queso de soja |
| 2 | 2 | Condimentos | Salsas dulces y picantes, delicias, comida par... |
| 3 | 4 | Lácteos | Quesos |
| 4 | 6 | Carnes | Carnes preparadas |
| 5 | 1 | Bebidas | Gaseosas, café, té, cervezas y maltas |
| 6 | 8 | Pescado/Marisco | Pescados, mariscos y algas |
| 7 | 3 | reaaaa | Postres, dulces y pan dulce |

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON

**Extraer los datos de las dos columnas en vectores**

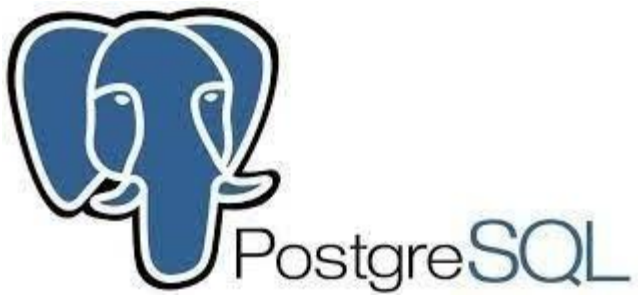
```
z0=x['NombreCategoría']
z1=x['Descripción']
```

**Insertar los datos de las dos columnas en vectores en la tabla CATEGORIA**

```
i=0
while i<len(x):
    sq="insert into
categoria(nombrecategoria,descripcion)  values(%s,%s)"
    sqv=(z0[i],z1[i])

    micursor.execute(sq,sqv)
    conn.commit()
    i=i+1
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRESQL CON PYTHON




pgAdmin File Object Tools Help

Browser

- Servers (1)
  - MiSena
    - Databases (2)
      - neptuno
        - Casts
        - Catalogs
        - Event Triggers
        - Extensions
        - Foreign Data Wrappers
        - Languages
        - Schemas (1)
          - public
            - Collations
            - Domains
            - FTS Configurations
            - FTS Dictionaries
            - FTS Parsers
            - FTS Templates
            - Foreign Tables
            - Functions
            - Materialized Views
            - Procedures
            - Sequences
            - Tables (1)
              - categorias
                - Columns
                - Constraints
                - Indexes
                - RLS Policies

Dashboard Properties SQL Statistics Dependencies Dependents public.categoria

No limit

public.categorias/neptuno/postgres@MiSena

Query Editor Query History

```
1 SELECT * FROM public.categorias
2 ORDER BY idcategoria ASC
```

Data Output Explain Messages Notifications

| | idcategoria [PK] integer | nombrecategoria text | descripcion text |
|---|---|---|---|
| 1 | 1 | Condimentos | Salsas dulces y ... |
| 2 | 2 | Condimentos | Salsas dulces y ... |
| 3 | 3 | Condimentos | Salsas dulces y ... |
| 4 | 4 | Condimentos | Salsas dulces y ... |
| 5 | 5 | Condimentos | Salsas dulces y ... |
| 6 | 6 | Condimentos | Salsas dulces y ... |
| 7 | 7 | Condimentos | Salsas dulces y ... |
| 8 | 8 | Condimentos | Salsas dulces y ... |

JOSE FERNANDO GALINDO SUAREZ

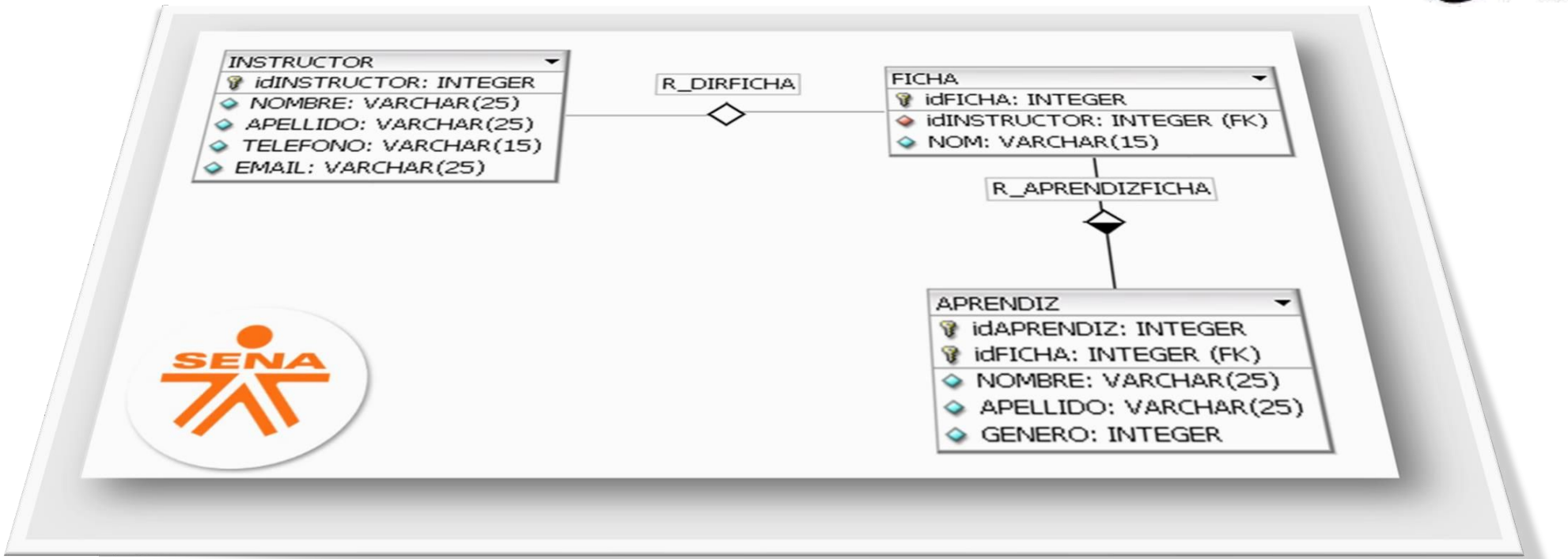
# CONECTANDO POSTGRESQL CON PYTHON

**Cerrar el cursor y la conexión**

```
micursor.close()
conn.close()
```

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA TRABAJO AUTONOMO POSTGRESQL CON PYTHON




JOSE FERNANDO GALINDO SUAREZ

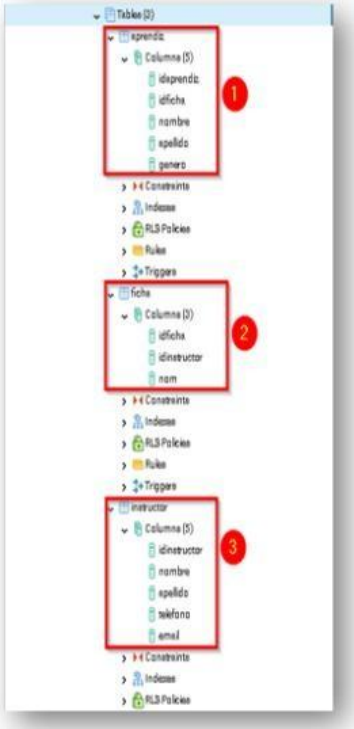
# PRACTICA POSTGRESQL CON PYTHON

**Esta práctica se deben crear las tablas (DDL) en pgAdmin de acuerdo al modelo relacional presentado:**

```
CREATE TABLE INSTRUCTOR (IDINSTRUCTOR SERIAL PRIMARY KEY,
                         NOMBRE VARCHAR(25),
                         APELLIDO VARCHAR(25),
                         TELEFONO VARCHAR(15),
                         EMAIL VARCHAR(25)
                        );
```

```
CREATE TABLE FICHA(IDFICHA SERIAL PRIMARY KEY,
        IDINSTRUCTOR INTEGER,
        NOM VARCHAR(25),
        CONSTRAINT INSTRUCTORFK FOREIGN KEY(IDINSTRUCTOR)
                        REFERENCES INSTRUCTOR(IDINSTRUCTOR));
```

```
CREATE TABLE APRENDIZ(IDAPRENDIZ INTEGER,
                      IDFICHA INTEGER,
                       NOMBRE VARCHAR(25),
                      APELLIDO VARCHAR(25),
                      GENERO INTEGER CHECK(GENERO IN(1,2)),
                      PRIMARY KEY(IDAPRENDIZ,IDFICHA) ,
                      CONSTRAINT FICHAFK FOREIGN KEY(IDFICHA)
                    REFERENCES FICHA(IDFICHA)
                                 );
```



JOSE FERNANDO GALINDO SUAREZ

# PRACTICA POSTGRESQL CON PYTHON

pgAdmin
Management Tools for PostgreSQL

**Ingresar los siguientes instructores:**

| NOMBRE | APELLIDO | TELEFONO | EMAIL |
|---|---|---|---|
| Uldarico | Andrade | 5941301 | uandrade@sena.edu.co |
| Erick | Granados | 31244455666 | egranados@sena.edu.co |
| Carlos | Patiño | 3106665544 | cpatiño@sena.edu.co |

**Ingresar las siguientes fichas:**

| IDINSTRUCTOR | NOMBRE |
|---|---|
| Erick Granados | 2687365 |
| Jose Fegasu | 2671743 |
| Carlos Patiño | 2465298 |

**Ingresar los siguientes aprendices:**

| IDAPRENDIZ | IDFICHA | NOMBRE | APELLIDO | GENERO |
|---|---|---|---|---|
| 1010113345 | 2687365 | ELVER | GALINDO | MASCULINO |
| 1011223456 | 2671743 | MARIA | GARCIA | FEMENINO |
| 78456723 | 2687365 | JUANITA | CORZO | FEMENINO |

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA POSTGRESQL CON PYTHON

pgAdmin
Management Tools for PostgreSQL

**Instalar las librerías necesarias**

```
!pip install psycopg2
```

**Importar las librerías**

```
import psycopg2
```

**Realizar la conexión hacia el servidor Postgres**

```
cnx=psycopg2.connect(database = "postgres",
            host="localhost",
            user="postgres",
            password="123",
            port="5432")
```

JOSE FERNANDO GALINDO SUAREZ

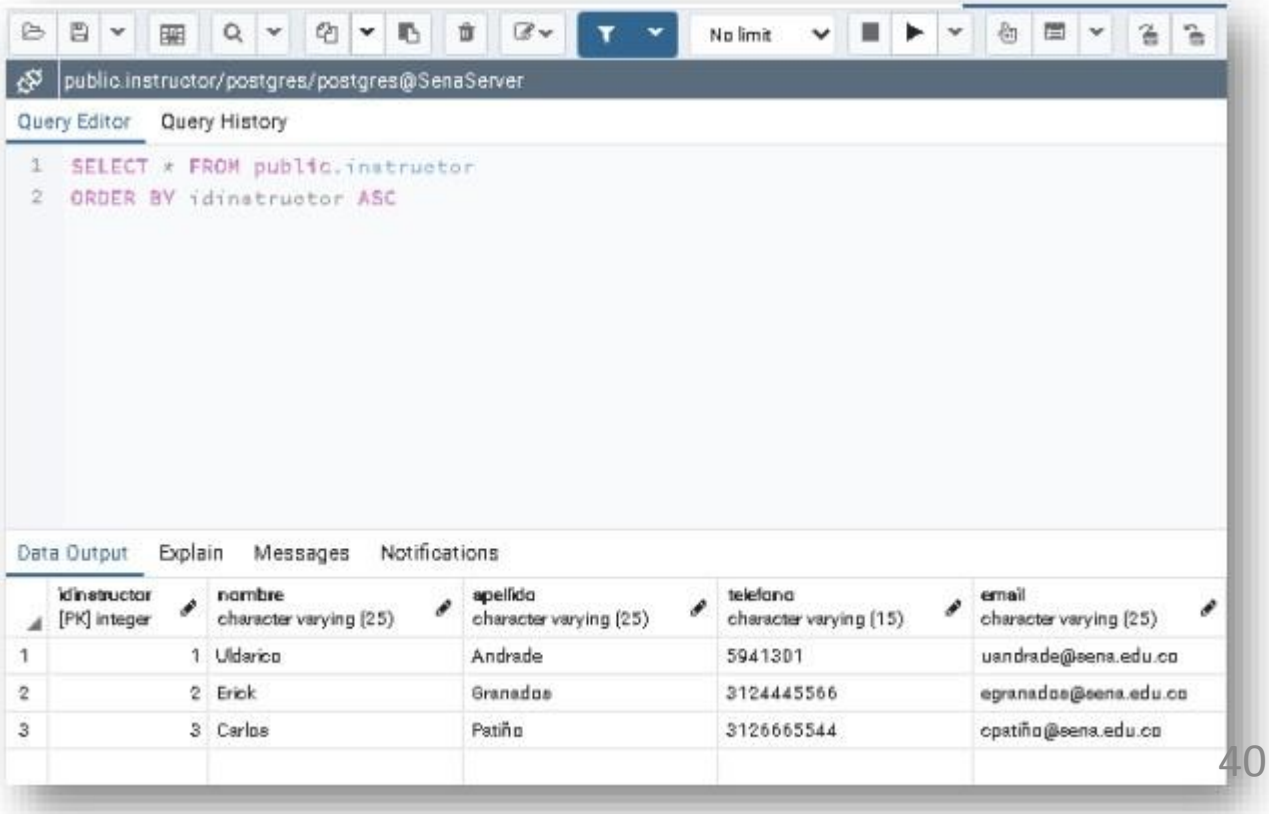
# PRACTICA POSTGRESQL CON PYTHON

pgAdmin
Management Tools for PostgreSQL

**Crear la sentencia INSERT de acuerdo a lo solicitado, ejecutar y confirmar la transacción.**

```
sql="insert into instructor(nombre,apellido,telefono,email)
values('Uldarico','Andrade','5941301','uandrade@sena.edu.co')"
cursor.execute(sql)
cnx.commit()
```

**Nota: Ingresar los siguientes instructores de acuerdo a lo solicitado Se verifica en POSTGRESQL**




| | idinstructor [PK] integer | nombre character varying (25) | apellido character varying (25) | telefono character varying (15) | email character varying (25) |
|---|---|---|---|---|---|
| 1 | 1 | Uldarico | Andrade | 5941301 | uandrade@sena.edu.co |
| 2 | 2 | Erick | Granados | 3124445566 | egranados@sena.edu.co |
| 3 | 3 | Carlos | Patiño | 3126665544 | cpatiño@sena.edu.co |

JOSE FERNANDO GALINDO SUAREZ

# TE RETO

# PRACTICA TRABAJO AUTONOMO CON PYTHON

1. Cargar la hoja de productos del libro Neptuno en una base de datos llamado “Ventas” en la tabla “productos”
2. Lo anterior se debe realizar en Python
3. Se debe repetir esta practica en Mysql, Postgres y sqlite utilizando SQL y ORM

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

PeeWee
Python library

# Object Relational Mapping (ORM)

**¿Qué es un ORM?**
**Es un modelo de programación que permite mapear las estructuras de una base de datos, sobre una estructura lógica de entidades con el objeto de simplificar y acelerar el desarrollo de las aplicaciones.**

## Ventajas de un ORM

- **Facilidad y velocidad de uso**
- **Abstracción de la base de datos usada.**
- **Seguridad de la capa de acceso a datos contra ataques.**
- **Reutilización. Permite utilizar los métodos de un objeto de datos desde distintas zonas de la aplicación, incluso desde aplicaciones distintas.**
- **Mantenimiento del código. Facilita el mantenimiento del código debido a la correcta ordenación de la capa de datos, haciendo que el mantenimiento del código sea mucho más sencillo.**

## Desventajas de un ORM

- **Lentitud en la carga de grandes volúmenes de datos al poner una capa más en el proceso puede mermar el rendimiento; en algunos casos es más rápido utilizar SQL puro.**
- **La curva de aprendizaje es mas lenta.**

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

| Field Type | Sqlite | Postgresql | MySQL |
|---|---|---|---|
| `AutoField` | integer | serial | integer |
| `BigAutoField` | integer | bigserial | bigint |
| `IntegerField` | integer | integer | integer |
| `BigIntegerField` | integer | bigint | bigint |
| `SmallIntegerField` | integer | smallint | smallint |
| `IdentityField` | not supported | int identity | not supported |
| `FloatField` | real | real | real |
| `DoubleField` | real | double precision | double precision |
| `DecimalField` | decimal | numeric | numeric |
| `CharField` | varchar | varchar | varchar |
| `FixedCharField` | char | char | char |
| `TextField` | text | text | text |
| `BlobField` | blob | bytea | blob |
| `BitField` | integer | bigint | bigint |
| `BigBitField` | blob | bytea | blob |
| `UUIDField` | text | uuid | varchar(40) |
| `BinaryUUIDField` | blob | bytea | varbinary(16) |
| `DateTimeField` | datetime | timestamp | datetime |
| `DateField` | date | date | date |
| `TimeField` | time | time | time |
| `TimestampField` | integer | integer | integer |
| `IPField` | integer | bigint | bigint |
| `BooleanField` | integer | boolean | bool |
| `BareField` | untyped | not supported | not supported |
| `ForeignKeyField` | integer | integer | integer |

- `null = False` – allow null values
- `index = False` – create an index on this column
- `unique = False` – create a unique index on this column. See also adding composite indexes.
- `column_name = None` – explicitly specify the column name in the database.
- `default = None` – any value or callable to use as a default for uninitialized models
- `primary_key = False` – primary key for the table
- `constraints = None` – one or more constraints, e.g. `[Check('price > 0')]`
- `sequence = None` – sequence name (if backend supports it)
- `collation = None` – collation to use for ordering the field / index
- `unindexed = False` – indicate field on virtual table should be unindexed (**SQLite-only**)
- `choices = None` – optional iterable containing 2-tuples of `value`, `display`
- `help_text = None` – string representing any helpful text for this field
- `verbose_name = None` – string representing the "user-friendly" name of this field
- `index_type = None` – specify a custom index-type, e.g. for Postgres you might specify a `'BRIN'` or `'GIN'` index.

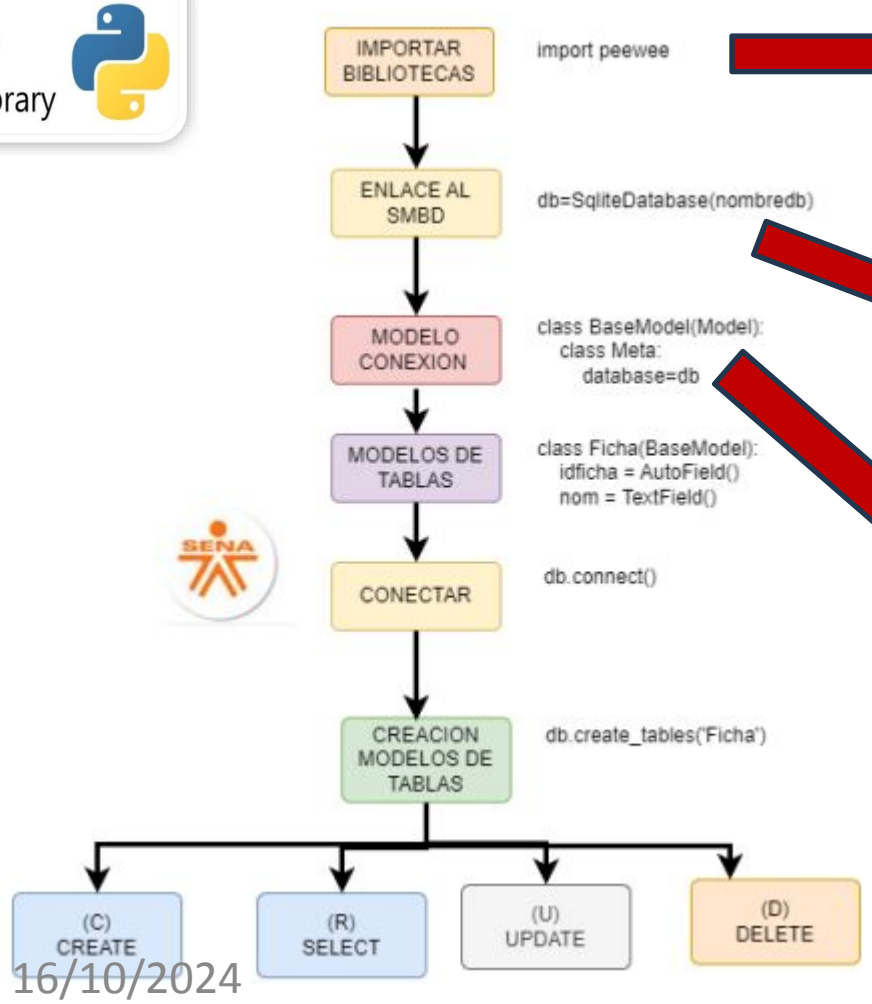
PeeWee
Python library

https://docs.peewee-orm.com/en/latest/peewee/models.html

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

PeeWee
Python library




**IMPORTAR BIBLIOTECAS**

```
#!pip install peewee
import peewee
from peewee import *
```

**ENLACE AL SMBD**

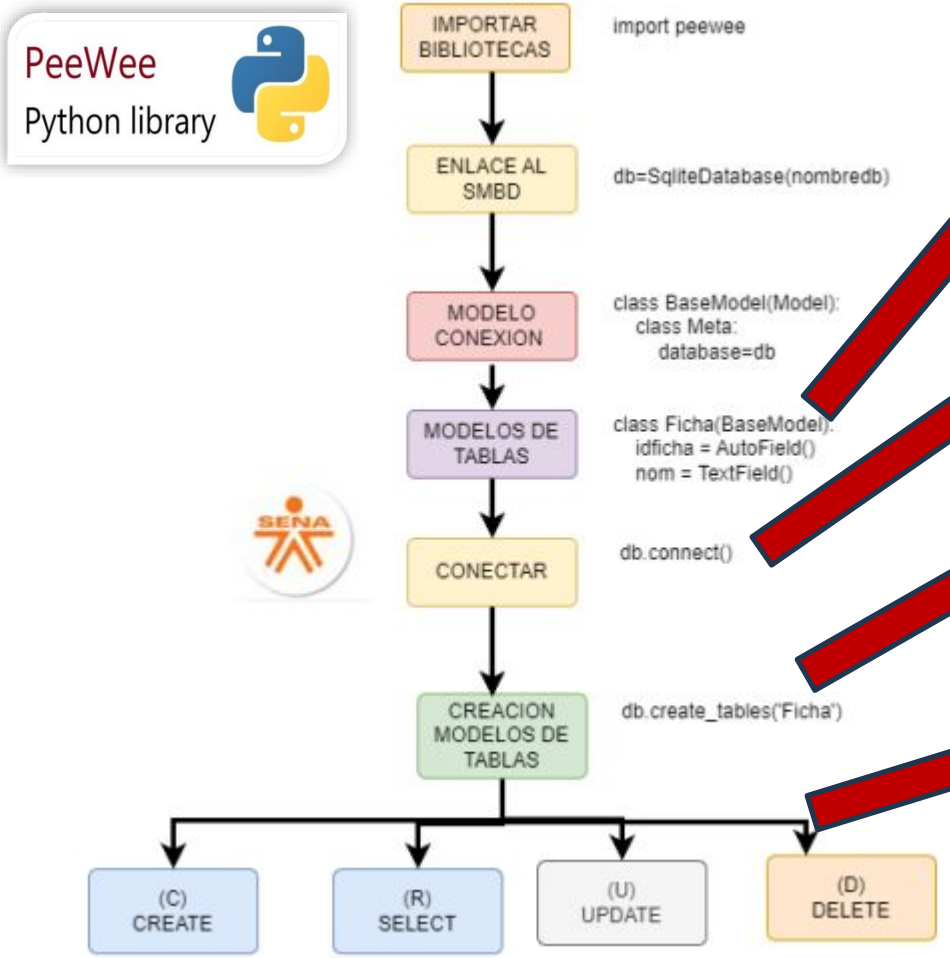
```
# Define database connection
conn = SqliteDatabase("./Neptuno.db")
```

**MODELO DE CONEXION**

```
class BaseModel(Model):
    class Meta:
        database=conn
```

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)




## Mapear la tabla CATEGORIA (modelo de tablas)

```
class categoria(BaseModel):
    idcategoria=AutoField(primary_key=True)
    nombrecategoria=TextField()
    descripcion=TextField()
```

## Conectar a la base de datos

```
conn.connect()
```

## Crear la base de datos

```
with conn:
    conn.create_tables([categoría])
```

## Matriz CRUD

```
#Inserta un registro
categoria.create(idcategoria=1,nombrecategoria='Pescado',descripcion='Atún')
#Actualiza un registro
categoria.update(descripcion='Atún Azul').where(categoria.idcategoria == 1).execute()
#Borra un registro
categoria.delete().where(categoria.idcategoria == 1).execute()
```

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

**Seleccionar todos los registros**

```
a=categoria.select()
for zz in a:
   print(zz.idcategoria,'|',zz.nombrecategoria,"|",zz.descripcion)
```

```
2 | Granos/Cereales | Pan, galletas, pasta y cereales
3 | Frutas/Verduras | Frutas secas y queso de soja
4 | Condimentos | Salsas dulces y picantes, delicias, comida para untar y aderezos
5 | Lácteos | Quesos
6 | Carnes | Carnes preparadas
7 | Bebidas | Gaseosas, café, té, cervezas y maltas
8 | Pescado/Marisco | Pescados, mariscos y algas
9 | reaaaa | Postres, dulces y pan dulce
```

PeeWee
Python library

JOSE FERNANDO GALINDO SUAREZ

# Object Relational Mapping (ORM)

**Seleccionar un registro**

```
a=categoria.select().where(categoria.idcategoria==5).get()
print(a.nombrecategoria,'|',a.descripcion)
```

**Lácteos | Quesos**

**Crear un registro**

```
categoria.create(nombrecategoria='Empanadas',descripcion='Horneadas')
```

**Actualizar un registro**

```
categoria.update({"descripcion":
                  "De Lechona"}).where(categoria.idcategoria==10).execute()
```

**Borrar un registro**

```
categoria.delete().where(categoria.idcategoria==10).execute()
```

PeeWee
Python library

JOSE FERNANDO GALINDO SUAREZ

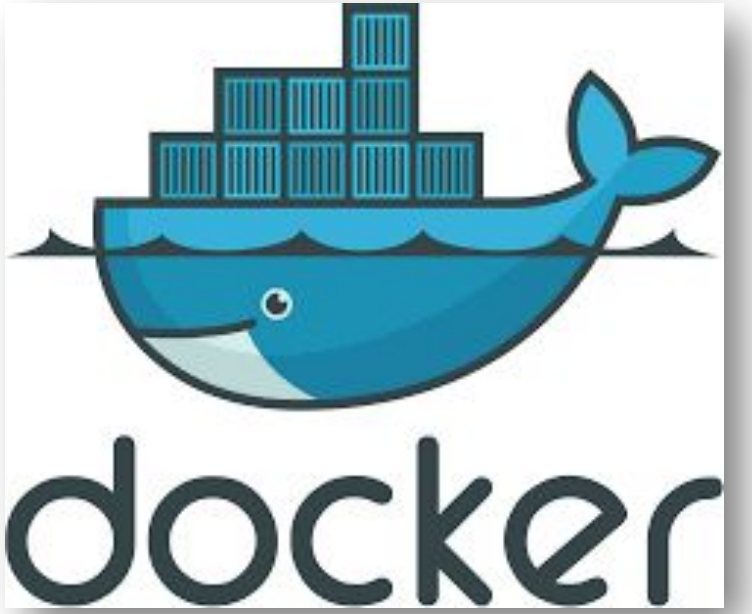
# DESPLIEGE DE APLICACIONES Y SERVICIOS EN CONTENEDORES DOCKER

| | |
|---|---|
| Modelo de Mejora Continua | LÍNEA TECNOLÓGICA DEL PROGRAMA<br>TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES GESTIÓN DE LA INFORMACIÓN<br>RED TECNOLÓGICA<br>TECNOLOGÍAS DE LA INFORMACIÓN, DISEÑO Y DESARROLLO DE SOFTWARE |

| DISEÑO DE ACCIONES DE FORMACION COMPLEMENTARIA | |
|---|---|
| CÓDIGO: | DENOMINACIÓN DEL PROGRAMA |
| 22810024 | DESPLIEGE DE APLICACIONES Y SERVICIOS EN CONTENEDORES DOCKER |
| VERSIÓN:<br>1 | SECTOR DEL PROGRAMA:<br>INDUSTRIA |
| Vigencia del Programa | Fecha inicio Programa: 15/06/2021<br>Fecha Fin Programa: El programa aún se encuentra vigente |

JOSE FERNANDO GALINDO SUAREZ

# ¿Qué es DOCKER?

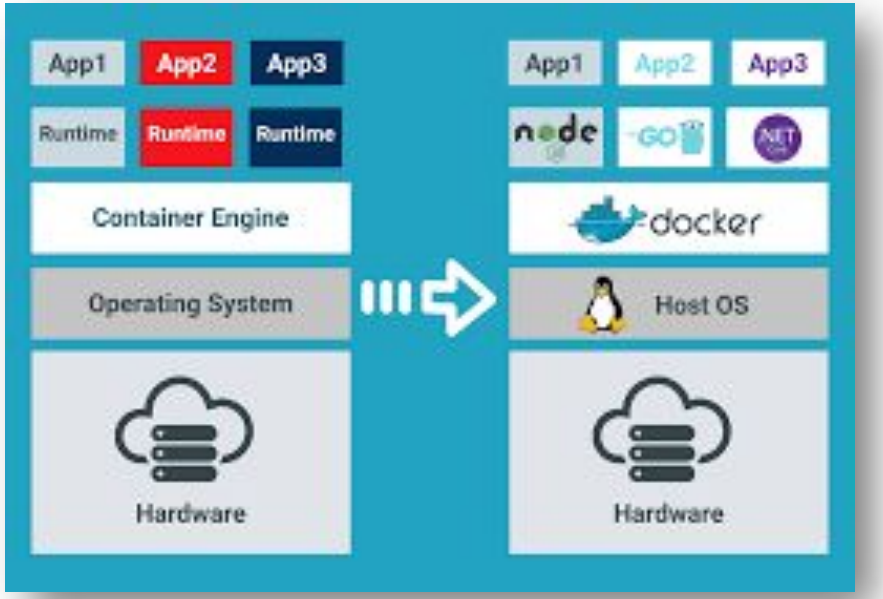



# ¿QUÉ ES DOCKER?

Es un software de código abierto utilizado para desplegar aplicaciones dentro de contenedores virtuales. La contenerización permite que varias aplicaciones funcionen en diferentes entornos complejos. Por ejemplo, Docker permite ejecutar el sistema de gestión de contenidos WordPress en sistemas Windows, Linux y macOS sin ningún problema.

JOSE FERNANDO GALINDO SUAREZ

# ¿QUÉ ES DOCKER?

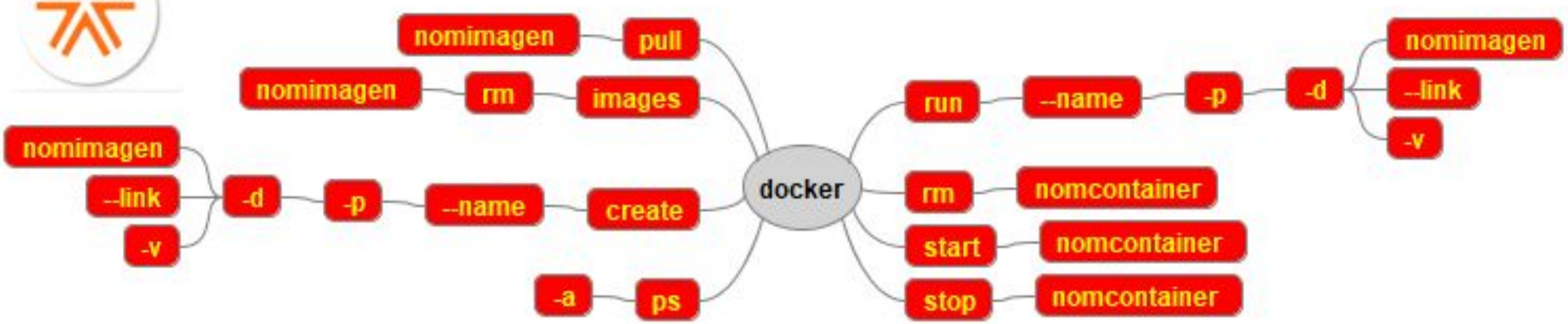



**La principal diferencia con las máquinas virtuales es que los contenedores Docker comparten el sistema operativo del anfitrión, mientras que las máquinas virtuales también tienen un sistema operativo invitado que se ejecuta sobre el sistema anfitrión. Este método de funcionamiento afecta al rendimiento, las necesidades de hardware y la compatibilidad con el sistema operativo.**

JOSE FERNANDO GALINDO SUAREZ

# COMANDOS DOCKER

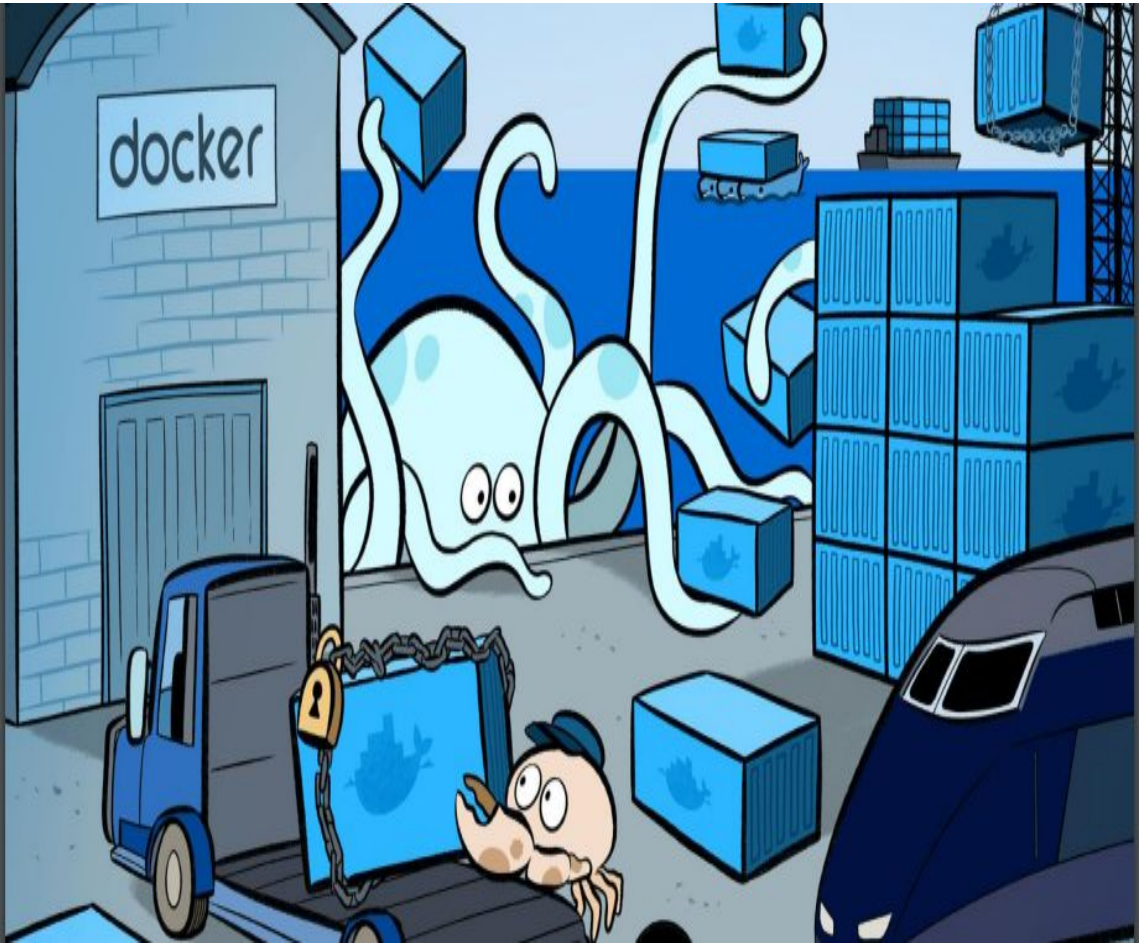
## COMANDOS DOCKER




Ver video de DOCKER

JOSE FERNANDO GALINDO SUAREZ

# COMANDOS DOCKER




```
docker run -d -p 4000:80 friendlyname#Run "friendlyname" mapping port 4000 to 80
docker container ls # List all running containers
docker container ls -a # List all containers, even those not running
docker container stop <hash> # Gracefully stop the specified container
docker container kill <hash> # Force shutdown of the specified container
docker container rm <hash> # Remove specified container from this machine
docker container rm $(docker container ls -a -q) # Remove all containers
docker image ls -a # List all images on this machine
docker image rm <image id> # Remove specified image from this machine
docker image rm $(docker image ls -a -q) # Remove all images from this machine
docker logs <containerName> # Shows the log of a container
```

JOSE FERNANDO GALINDO SUAREZ

# PRACTICA DOCKER

**PRACTICA CON MYSQL Y DOCKER**
**Instalar Docker Desktop para poder desarrollar esta práctica.**
**Crear el contenedor con la imagen de Mariadb (el comando debe ser en solo una línea)**
**docker run -p 3306:3306 --name mysqldb -e MYSQL_ROOT_PASSWORD=root-d mysql**

**Crear un contenedor para PHP**
**docker run -p 9902:80 --name miservidor -v d:/Docker/paginas:/var/www/html --link mysqldb -d php:8.0-apache**

**Crear el contenedor con la imagen de PhpMyAdmin Mariadb (el comando debe ser en solo una línea)**
**docker run -p 8080:80 --name miadmin -e PMA_HOST=mysqldb -d phpmyadmin/phpmyadmin**

**Se ingresa a la línea de comando bash para iniciar a mysql**
**docker exec -it e7c7e bash**

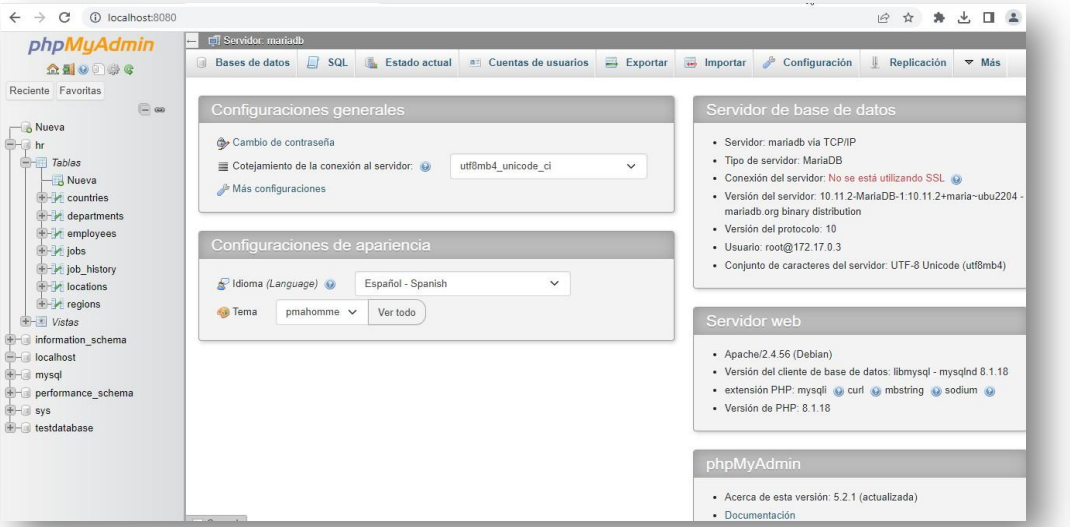
JOSE FERNANDO GALINDO SUAREZ

# PRACTICA DOCKER

## Ingresar a mysql

```
root@a906fa108b95: /
root@a906fa108b95:/# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 27
Server version: 10.11.2-MariaDB-1:10.11.2+maria~ubu2204 mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> use hr
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MariaDB [hr]> show tables;
+------------------+
| Tables_in_hr     |
+------------------+
| countries        |
| departments      |
| emp_details_view |
| employees        |
| job_history      |
| jobs             |
| locations        |
| regions          |
+------------------+
8 rows in set (0.000 sec)

MariaDB [hr]>
```

## Ingresar al PhpMyAdmin desde http://localhost:8080/




JOSE FERNANDO GALINDO SUAREZ

# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK

## Crear el archivo “routes.py”

```
from flask import Flask
from flask_cors import CORS
def create_app():
    app=Flask(__name__)
    CORS(app)
    return app

app = create_app() # CREATE THE FLASK APP
@app.route("/")
def inicio():
    return "CONDOMINIO PALMANOVA SINSONTE"
if __name__ == '__main__':
    app.run(host='0.0.0.0',port=5000)
```

## Crear el archivo “Dockerfile”

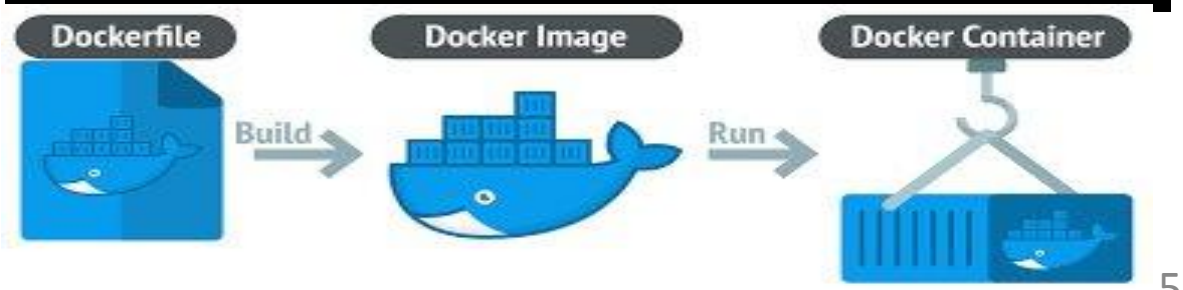
```
FROM python:3.7
WORKDIR /app
 COPY requi.txt .
RUN pip install -r requi.txt
COPY . .
ENTRYPOINT [ "python" ]
CMD [ "routes.py" ]
```

## Crear el archivo “requi.txt”

```
flask
flask-cors
requests
mongo
```

## Se ejecuta en la línea de comandos

**docker build -t sinsonte .**
**docker run -p 5000:5000 --name apisinsonte sinsonte**
**docker start apisinsonte**




JOSE FERNANDO GALINDO SUAREZ

# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK

**docker build -t sinsonte .**

```
C:\CPSI\SINSONTE>docker build -t sinsonte .
[+] Building 2.3s (9/9) FINISHED                                    docker:desktop-linux
 => [internal] load build definition from Dockerfile                                0.3s
 => => transferring dockerfile: 187B                                                0.1s
 => [internal] load metadata for docker.io/library/python:3.7                       0.1s
 => [internal] load .dockerignore                                                   0.3s
 => => transferring context: 2B                                                     0.2s
 => [internal] load build context                                                   0.3s
 => => transferring context: 89B                                                    0.1s
 => [1/4] FROM docker.io/library/python:3.7                                         0.0s
 => CACHED [2/4] COPY . /app                                                        0.0s
 => CACHED [3/4] WORKDIR /app                                                       0.0s
 => CACHED [4/4] RUN pip install -r requi.txt                                       0.0s
```

**docker run -p 5000:5000 --name apisinsonte sinsonte**

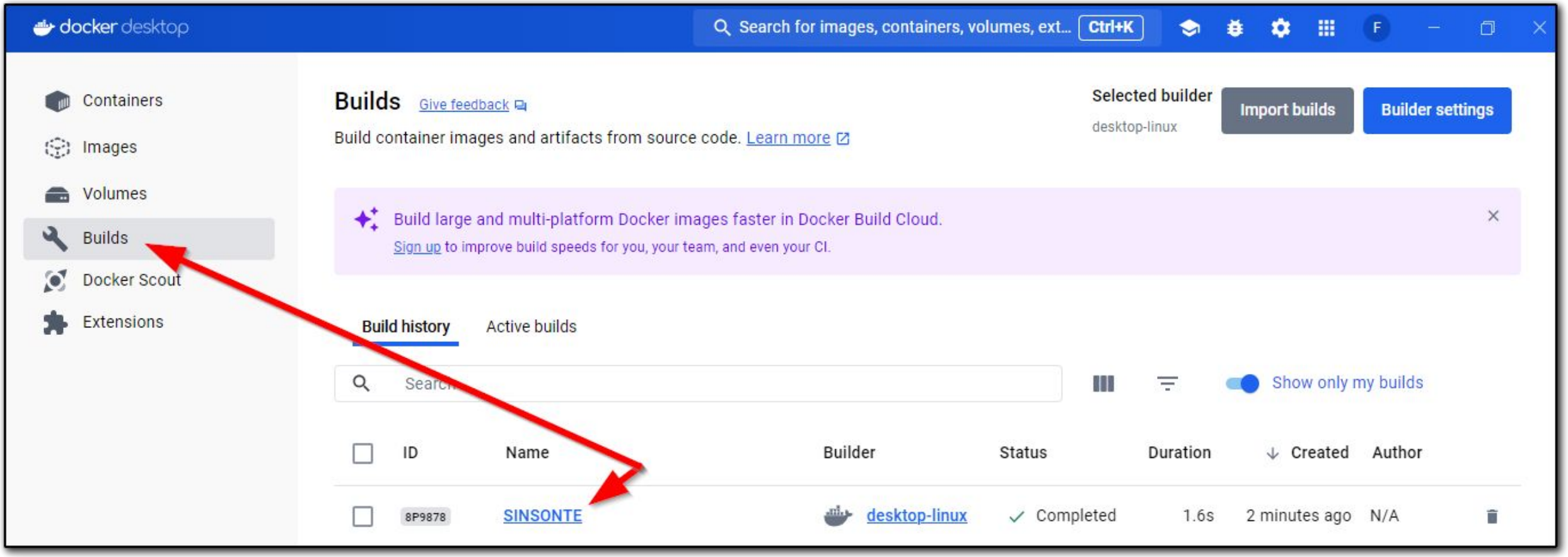
```
C:\CPSI\SINSONTE>docker run -p 5000:5000 --name apisinsonte sinsonte
 * Serving Flask app 'routes'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:5000
 * Running on http://172.17.0.2:5000
Press CTRL+C to quit
```

**docker start apisinsonte**

JOSE FERNANDO GALINDO SUAREZ

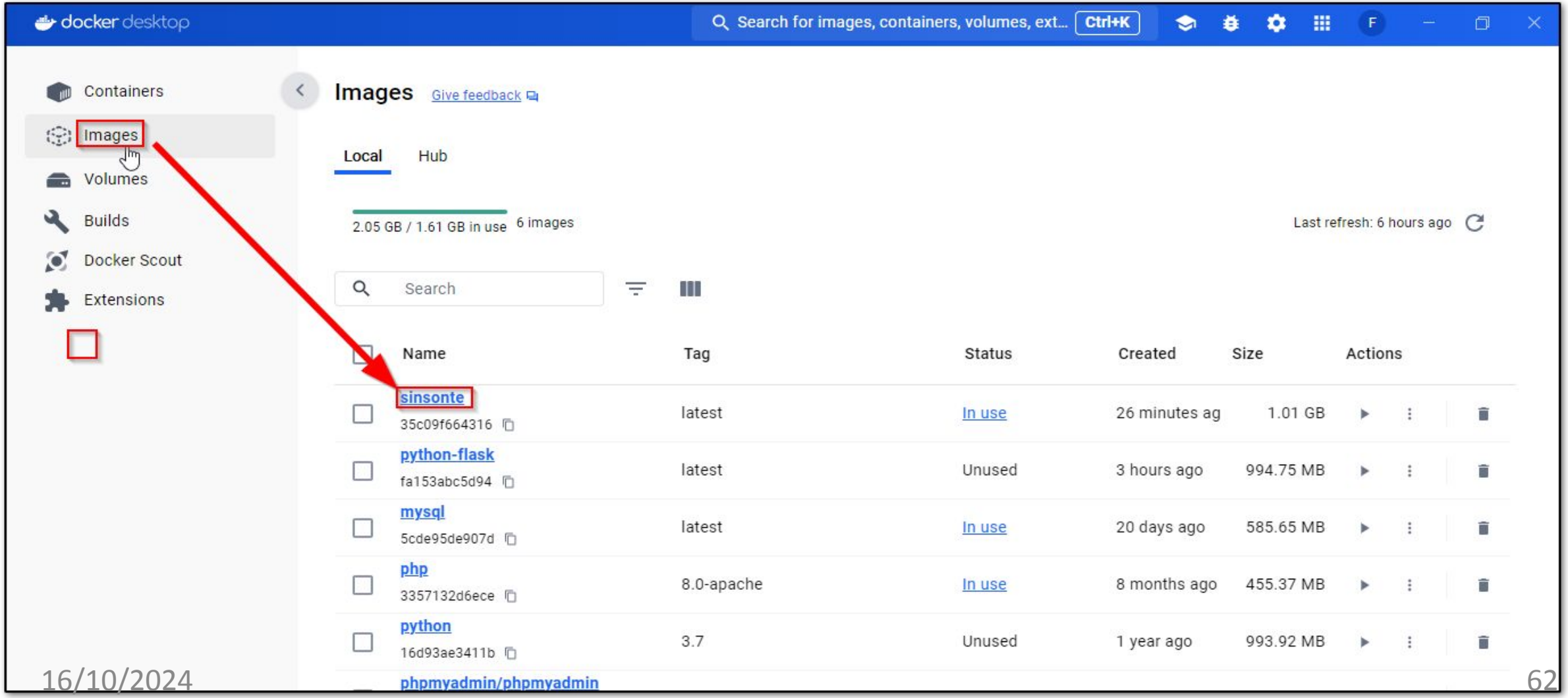
# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK

**Build .**



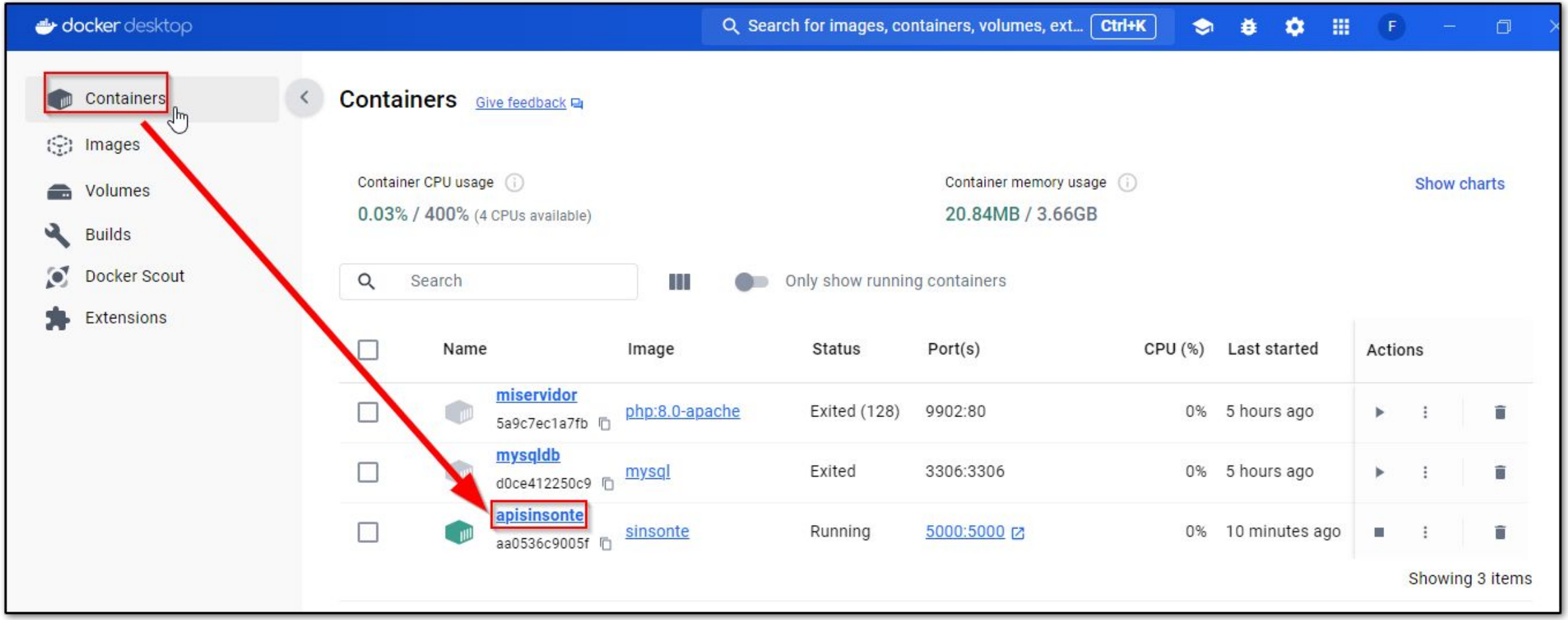

JOSE FERNANDO GALINDO SUAREZ

# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK




JOSE FERNANDO GALINDO SUAREZ

# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK




JOSE FERNANDO GALINDO SUAREZ

# CREAR UN CONTENEDOR DOCKER PARA UNA APLICACIÓN FLASK

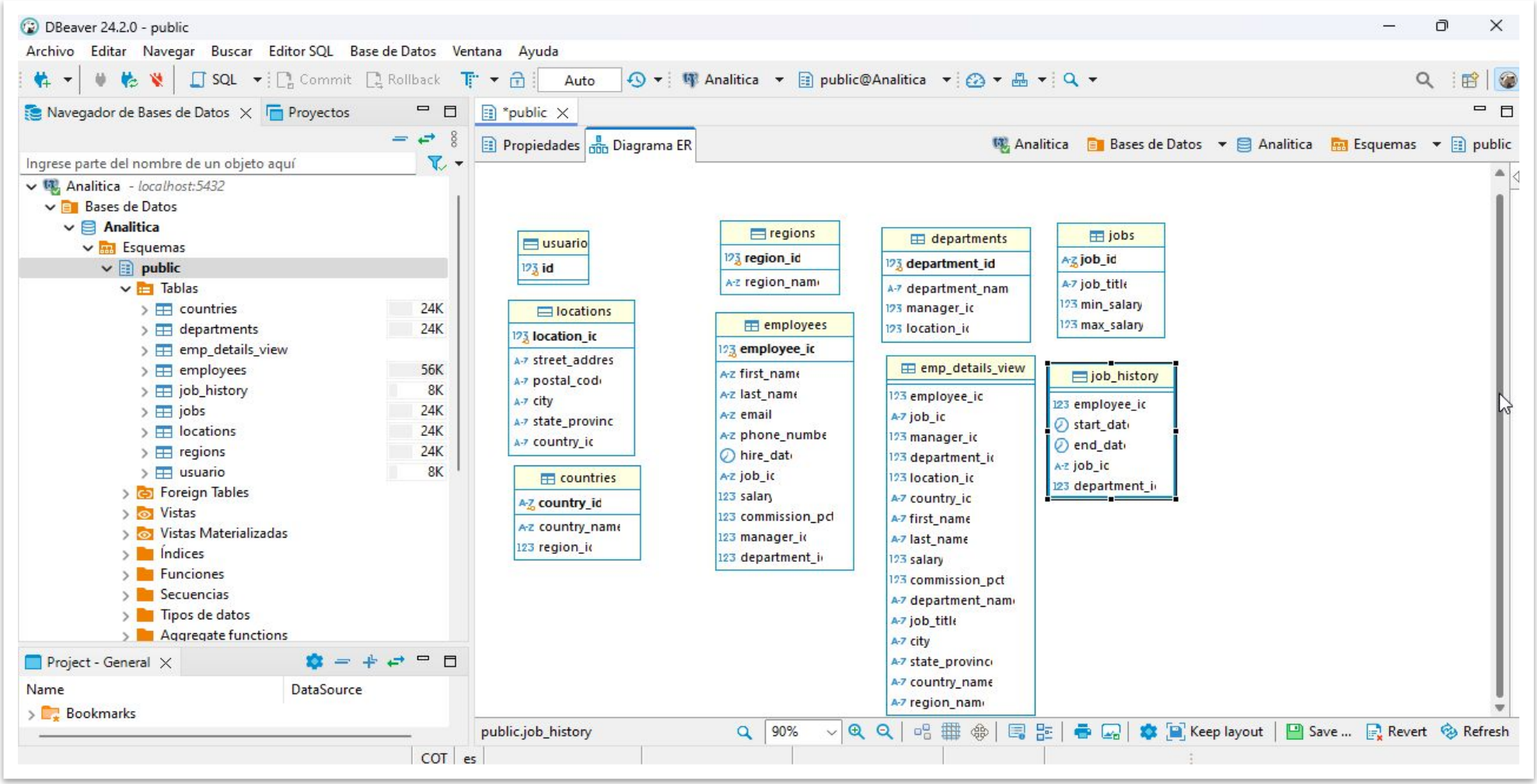
JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRES CON DOCKER

```
docker pull postgres
docker run -e POSTGRES_PASSWORD=root -e POSTGRES_USER=root -e
POSTGRES_DB=Analitica -p 5432:5432 --name pgsqlsena -d postgres
docker exec -it pgsqlsena bash
psql -U root --db Analitica --password

CREATE TABLE USUARIO(ID SERIAL PRIMARY KEY);
https://github.com/fegasu/tcprogAnalitica/blob/main/DATASET/HRPOSTGRESQL.sql
```

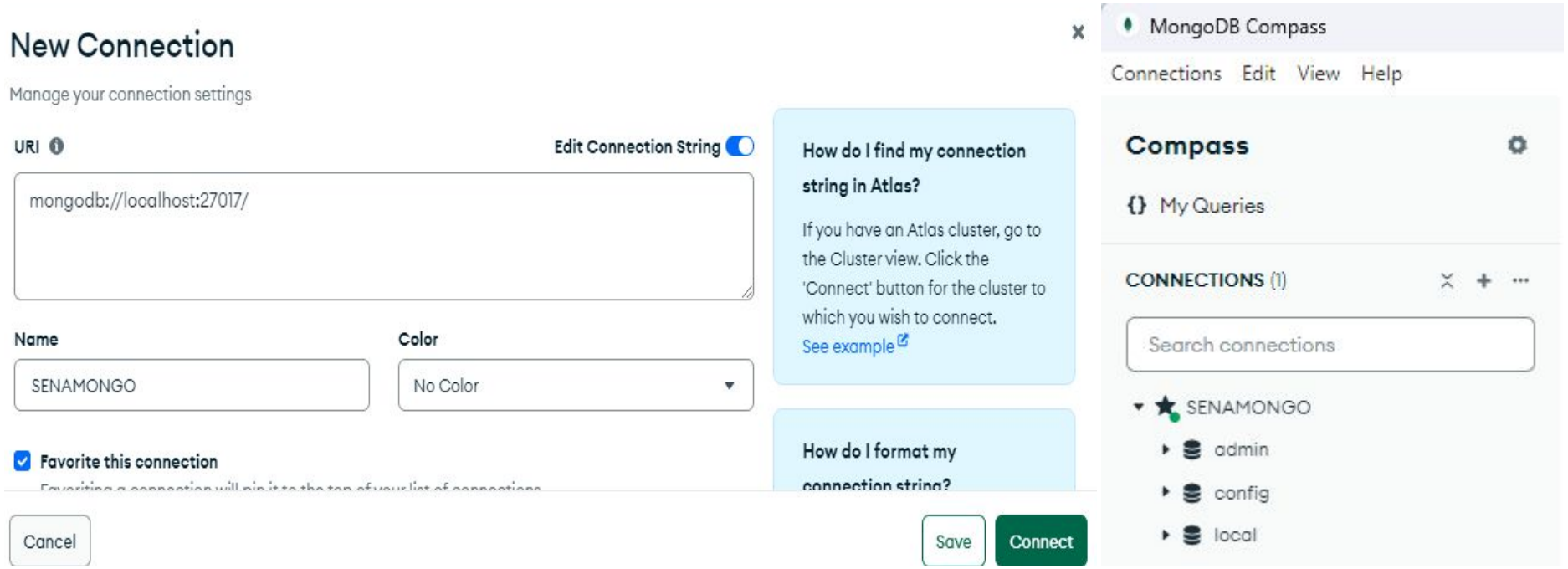
JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO POSTGRES CON DOCKER



# CONECTANDO MONGO CON DOCKER

```
docker pull mongo
docker run -p 27017:27017 --name mongosena -d mongo
```




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MYSQL CON DOCKER

```
docker pull mariadb
docker run -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root --name mariadbsena -d mariadb
docker exec -it mariadbsena bash
mysql -u root -p
CREATE DATABASE Analitica;
use Analitica
CREATE TABLE USUARIO(ID INTEGER PRIMARY KEY AUTO_INCREMENT);
```

```
root@5e1d8fdb115a: /

C:\Users\FEGASU>docker pull mariadb  1
Using default tag: latest
latest: Pulling from library/mariadb
31e907dcc94a: Pull complete
5f1ca115b417: Pull complete
5f79816720d8: Pull complete
1a674d3e0d95: Pull complete
2f5436342963: Pull complete
11d8a022d3df: Pull complete
a0478840c499: Pull complete
ecc1ceb4f1da: Pull complete
Digest: sha256:346f3a6792bf88014a9d9290927c12c13522a50e0e7e6d40aaa9add73df0bcc6
Status: Downloaded newer image for mariadb:latest
docker.io/library/mariadb:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview mariadb

C:\Users\FEGASU>docker run -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root --name mariadbsena -d mariadb  2
5e1d8fdb115a4d10906174ffcde6e66d1acc3b64145c0868cee2899e34af4388

C:\Users\FEGASU>docker exec -it mariadbsena bash  3
root@5e1d8fdb115a:/#  4
```

HR DATASET

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MYSQL CON DOCKER

**HR DATASET**

docker desktop PERSONAL | Search for images, containers, volume... Ctrl+K

- Containers 1
- Images
- Volumes
- Builds
- Docker Scout
- Extensions

## Containers

Give feedback

Container CPU usage: 0.02% / 1200% (12 CPUs available)

Container memory usage: 193.1MB / 30.43GB

Show charts

Search | Only show running containers

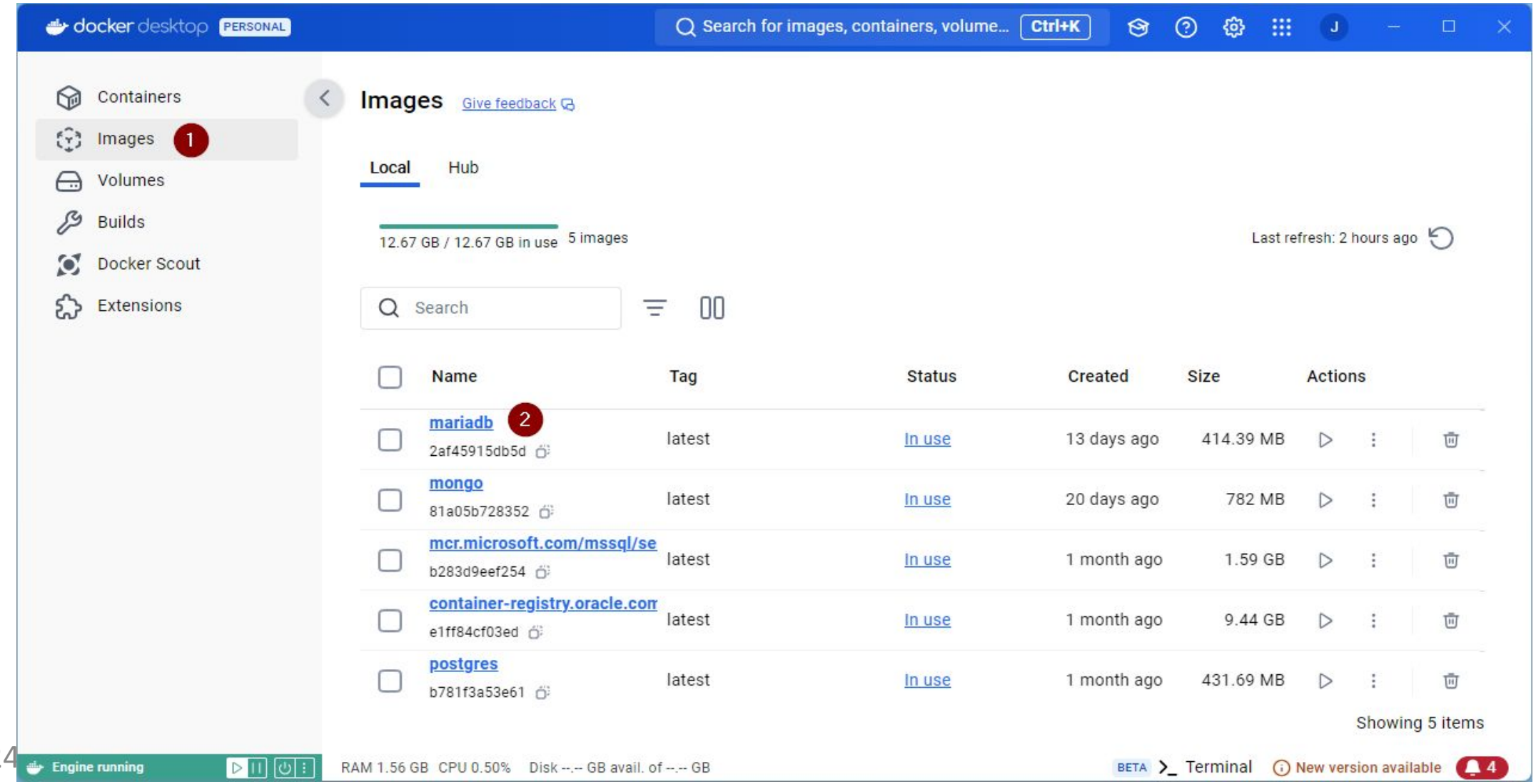
| Name | Image | Status | Port(s) | CPU (%) | Last started | Actions |
|---|---|---|---|---|---|---|
| OracleSen 39902591f5 | container-registry.oracle.cc | Exited (255) | 1521:1521 | 0% | 20 hours ago | |
| misqlserve 5c0ad43a91 | mcr.microsoft.cc | Exited (255) | 1433:1433 | 0% | 1 day ago | |
| mongosen 75dabf7c6d | mongo | Exited | 27017:27017 | 0% | 1 hour ago | |
| pgsqlsena e59f65ccb4 | postgres | Exited | 5432:5432 | 0% | 2 hours ago | |
| mariadbse 5e1d8fdb11 (2) | mariadb | Running | 3306:3306 | 0.01% | 5 minutes ago | |

Showing 5 items

Engine running | RAM 1.54 GB CPU 1.16% Disk --.-- GB avail. of --.-- GB | BETA Terminal | New version available 4

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MYSQL CON DOCKER

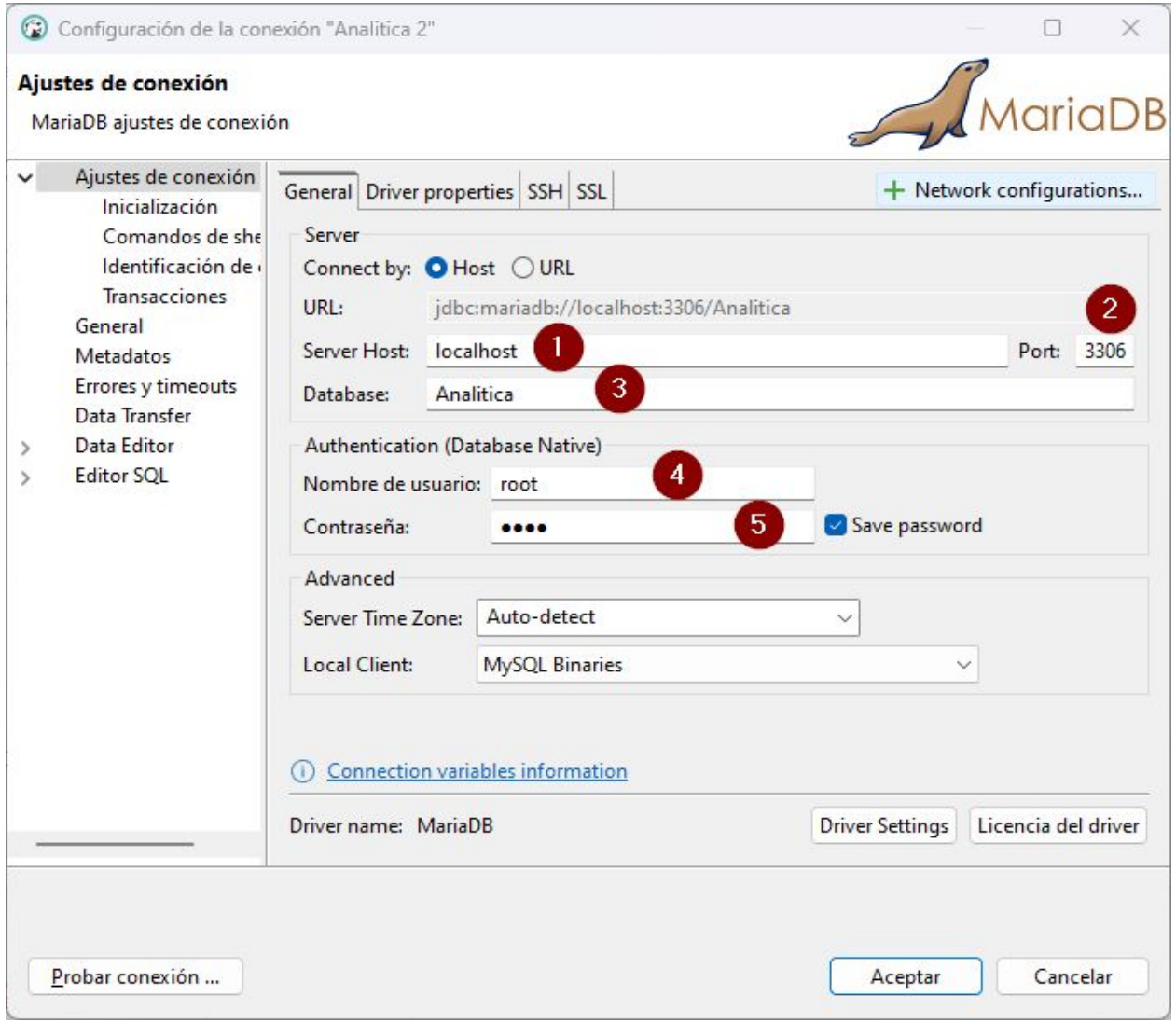



HR DATASET



JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MYSQL CON DOCKER



HR DATASET

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO MYSQL CON DOCKER

HR DATASET

DBeaver 24.2.0 - <Analitica 2> Script-2

Archivo Editar Navegar Buscar Editor SQL Base de Datos Ventana Ayuda

SQL (1) Commit Rollback Auto Analitica 2 Analitica

Navegador de ... Proyectos

Ingrese parte del nombre de un objeto aquí

- Analitica - localhost:5432
- Analitica 2 - localhost:3306
  - Databases
    - sys
    - Analitica (6)
      - Tables (7)
        - USUARIO 16K
      - Views
      - Indexes
      - Procedures
      - Packages
      - Sequences
      - Triggers
      - Events
  - Users
  - Administer
  - System Info
- free - localhost:1521
- master - localhost:1433

Analitica | *<Analitica 2> Script-2 (2)

```
CREATE DATABASE Analitica; (3)
use Analitica (4)

CREATE TABLE USUARIO(ID INTEGER PRIMARY KEY AUTO_INCREMENT); (5)
```

Estadísticas 1

| Name | Value |
|---|---|
| Updated Rows | 0 |
| Query | CREATE TABLE USUARIO(ID INTEGER PRIMARY KEY AUTO_INCREMENT) |
| Start time | Sun Sep 15 17:19:26 COT 2024 |
| Finish time | Sun Sep 15 17:19:27 COT 2024 |

Project - General

| Name | DataSource |
|---|---|
| Bookmarks | |
| Dashboards | |

COT es Editable Inserción inteligente 3 : 1 : 43

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO SQLSERVER CON DOCKER

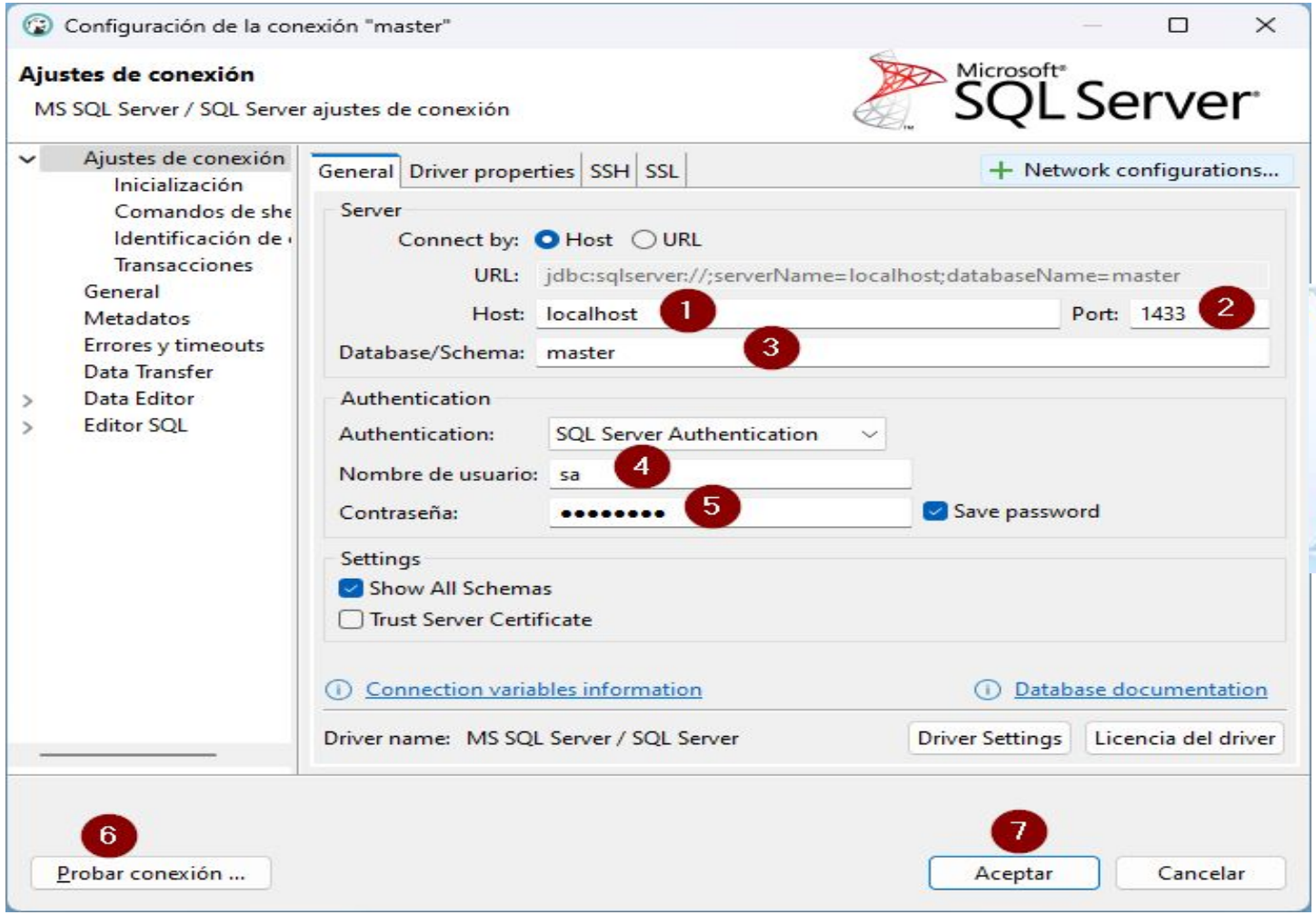
```
docker pull mcr.microsoft.com/mssql/server
docker scout quickview mcr.microsoft.com/mssql/server
docker run --name "misqlserver" -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=SQL#1234" -p
1433:1433 -d mcr.microsoft.com/mssql/server
docker start misqlserver
```

```
C:\WINDOWS\system32\cmd.

C:\Users\FEGASU>docker scout quickview mcr.microsoft.com/mssql/server  1
    i New version 1.13.0 available (installed version is 1.11.0) at https://github.com/docker/scout-cli
    v Image stored for indexing
    v Indexed 264 packages
    ! failed to delete temporary image archive C:\Users\FEGASU\AppData\Local\Temp\docker-scout\sha256\b283d9eef2544af92aeb38f8bb3d55ce17de
037eafb04eeeb3ee2c52033476ec\31859fb4-8a2b-4d98-be49-411ad4772a0a: remove C:\Users\FEGASU\AppData\Local\Temp\docker-scout\sha256\b283d9eef
2544af92aeb38f8bb3d55ce17de037eafb04eeeb3ee2c52033476ec\31859fb4-8a2b-4d98-be49-411ad4772a0a: The process cannot access the file because i
t is being used by another process.
    i Base image was auto-detected. To get more accurate results, build images with max-mode provenance attestations.
      Review docs.docker.com ↗ for more information.

  Target                  | mcr.microsoft.com/mssql/server:latest |  0C   0H   10M   15L
    digest                | b283d9eef254                          |
  Base image              | ubuntu:22.04                          |  0C   0H    6M   16L
  Refreshed base image    | ubuntu:22.04                          |  0C   0H    4M   12L
                                                                  |                -2    -4
  Updated base image      | ubuntu:24.04                          |  0C   0H    3M    3L
                                                                  |                -3   -13

What's next:
    View vulnerabilities → docker scout cves mcr.microsoft.com/mssql/server
    View base image update recommendations → docker scout recommendations mcr.microsoft.com/mssql/server
    Include policy results in your quickview by supplying an organization → docker scout quickview mcr.microsoft.com/mssql/server --org <o
rganization>


C:\Users\FEGASU>docker run --name "misqlserver" -e "ACCEPT_EULA=Y" -e "MSSQL_SA_PASSWORD=SQL#1234" -p 1433:1433 -d mcr.microsoft.com/mssql
/server  2
5b6dd34eb0dd408c8ffc3e0466bd69e84fd30daf9c0ecf9656d8f02a684c4009

C:\Users\FEGASU>docker start misqlserver  3
misqlserver  4
```

HR DATASET

JOSE FERNANDO GALINDO SUAREZ

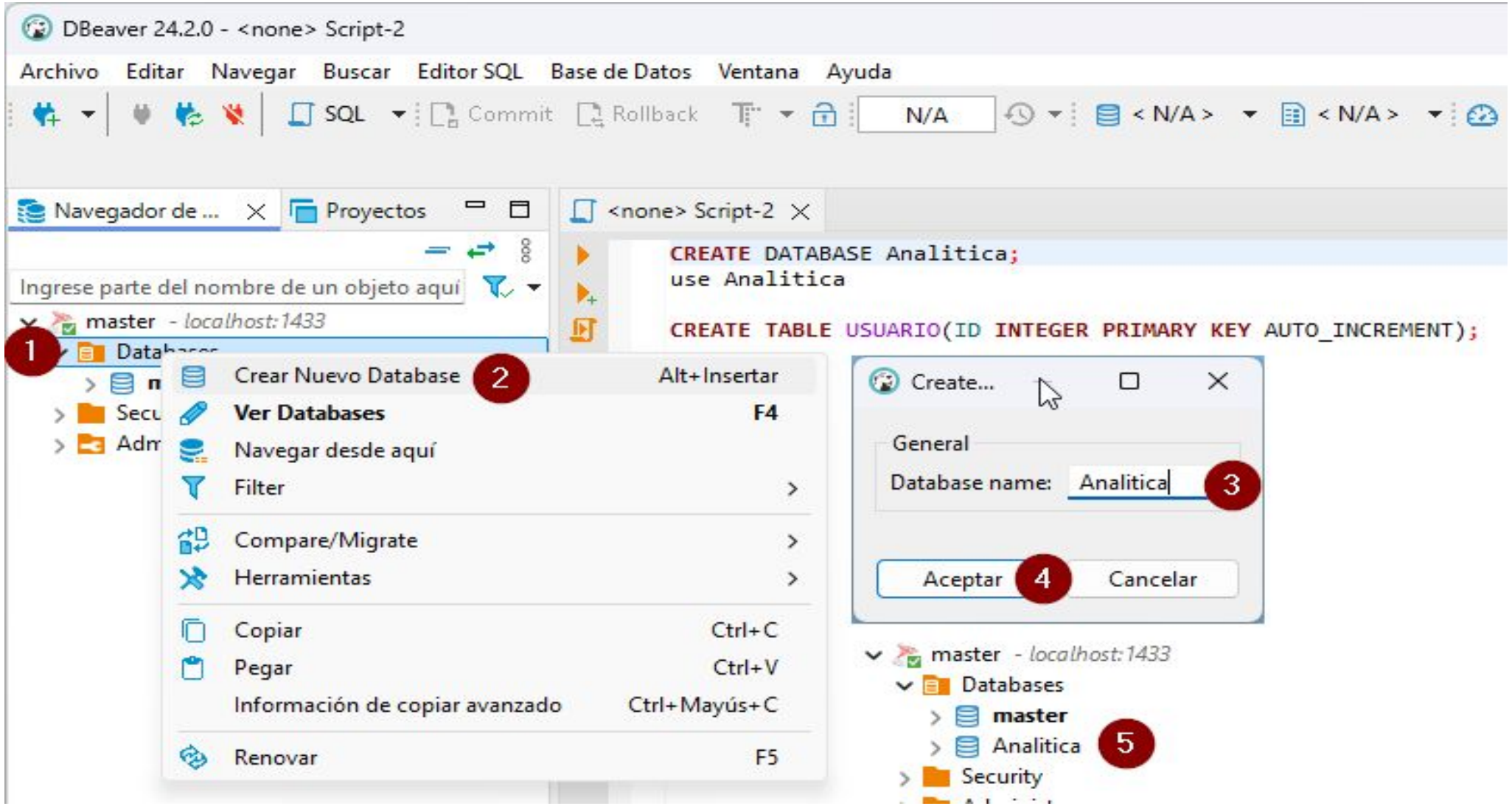
# CONECTANDO SQLSERVER CON DOCKER



HR DATASET

JOSE FERNANDO GALINDO SUAREZ

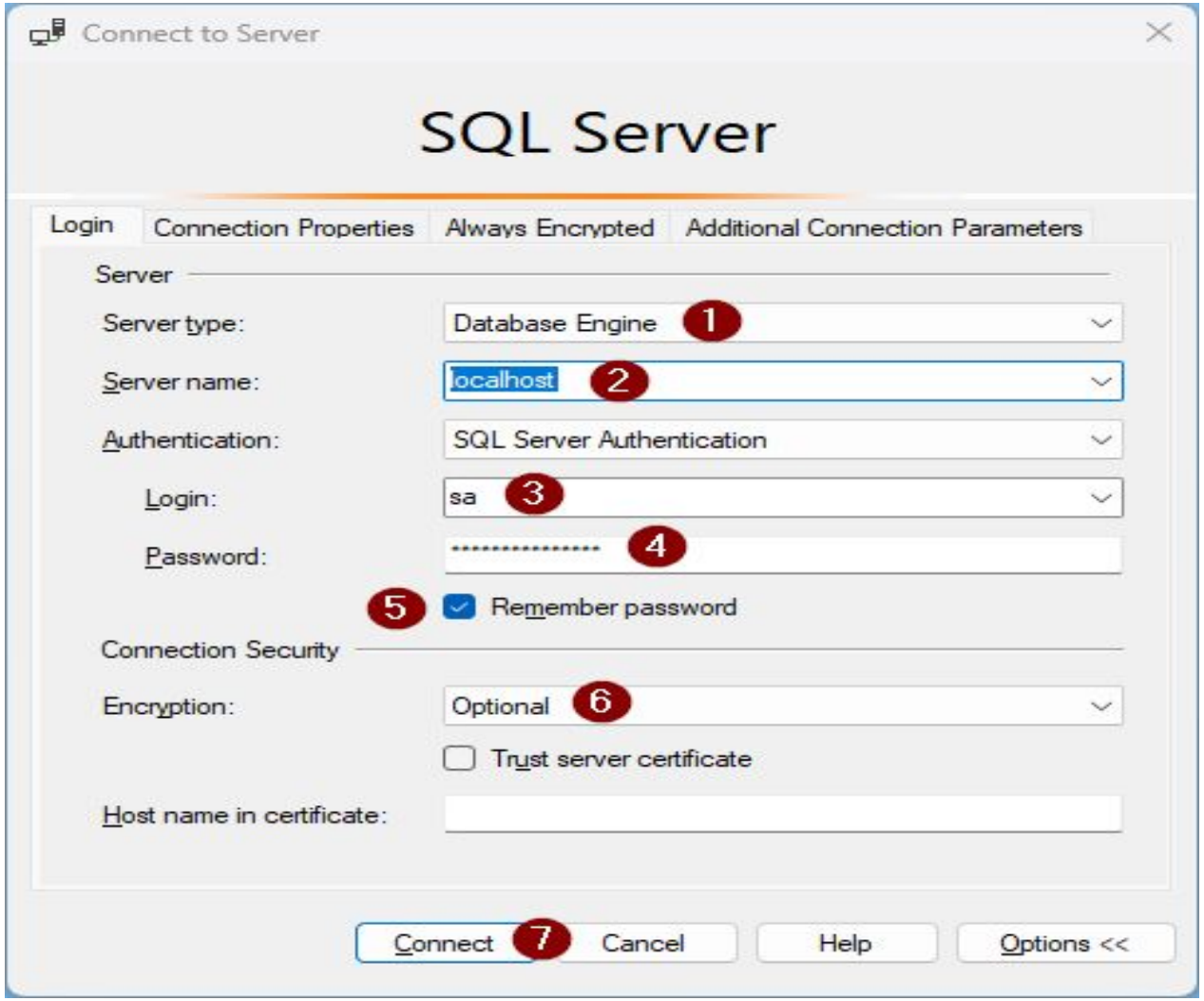
# CONECTANDO SQLSERVER CON DOCKER



**HR DATASET**

**SQLSERVER MANAGEMETE STUDIO**

# CONECTANDO SQLSERVER CON DOCKER

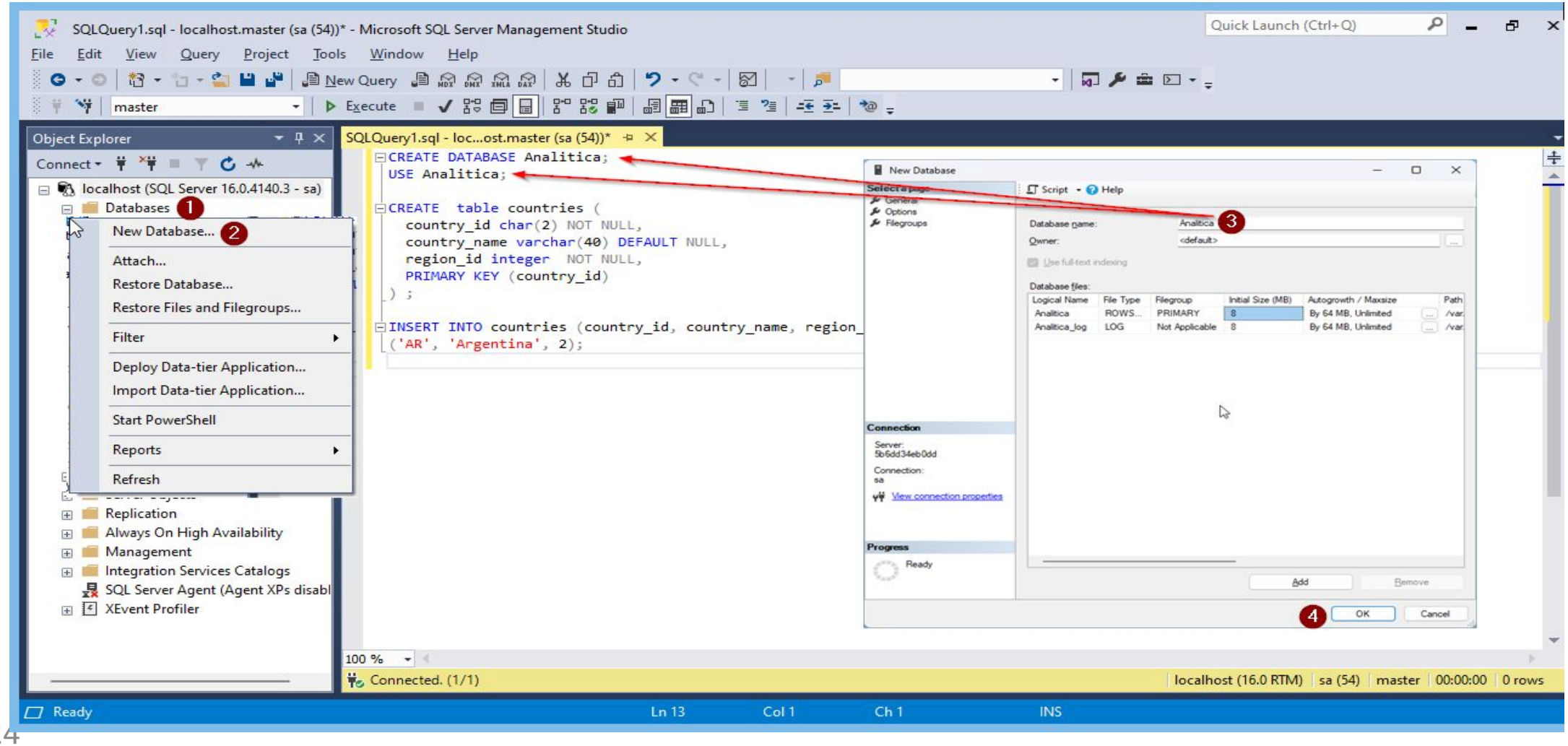


**HR DATASET**

**SQLSERVER MANAGEMETE STUDIO**

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO SQLSERVER CON DOCKER




**HR DATASET**

**SQLSERVER MANAGEMETE STUDIO**

# CONECTANDO ORACLE CON DOCKER

```
Docker pull container-registry.oracle.com/database/free
Docker run -d --name "OracleSena" -p 1521:1521 -e
"ORACLE_PWD=ora1234"
container-registry.oracle.com/database/free
docker start container-registry.oracle.com/database/free
```

HR DATASET

JOSE FERNANDO GALINDO SUAREZ

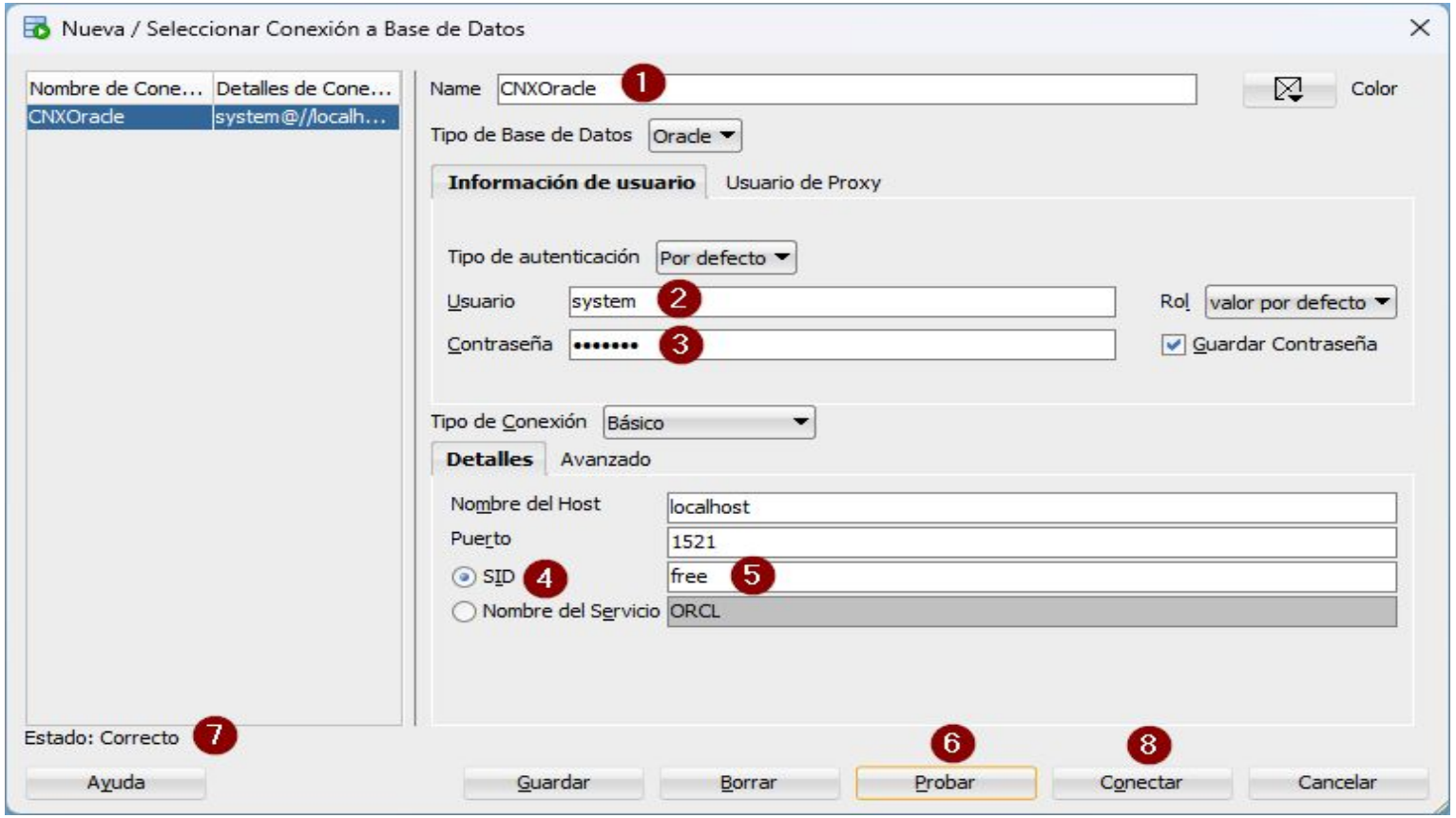
# CONECTANDO ORACLE CON DOCKER

ORACLE DATABASE

docker

SENA

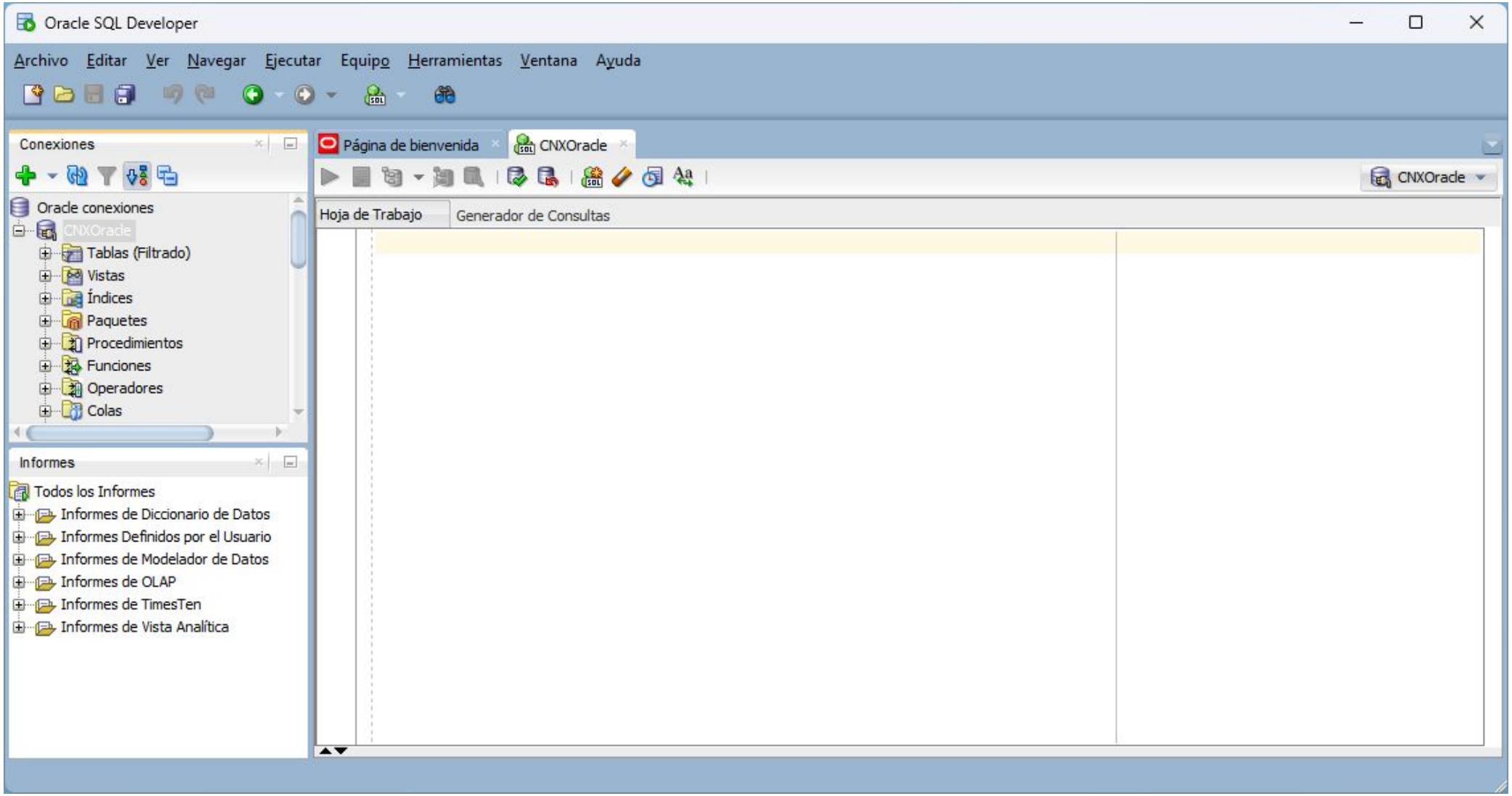
**HR DATASET**



JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO ORACLE CON DOCKER

**HR DATASET**




JOSE FERNANDO GALINDO SUAREZ

# TE RETO

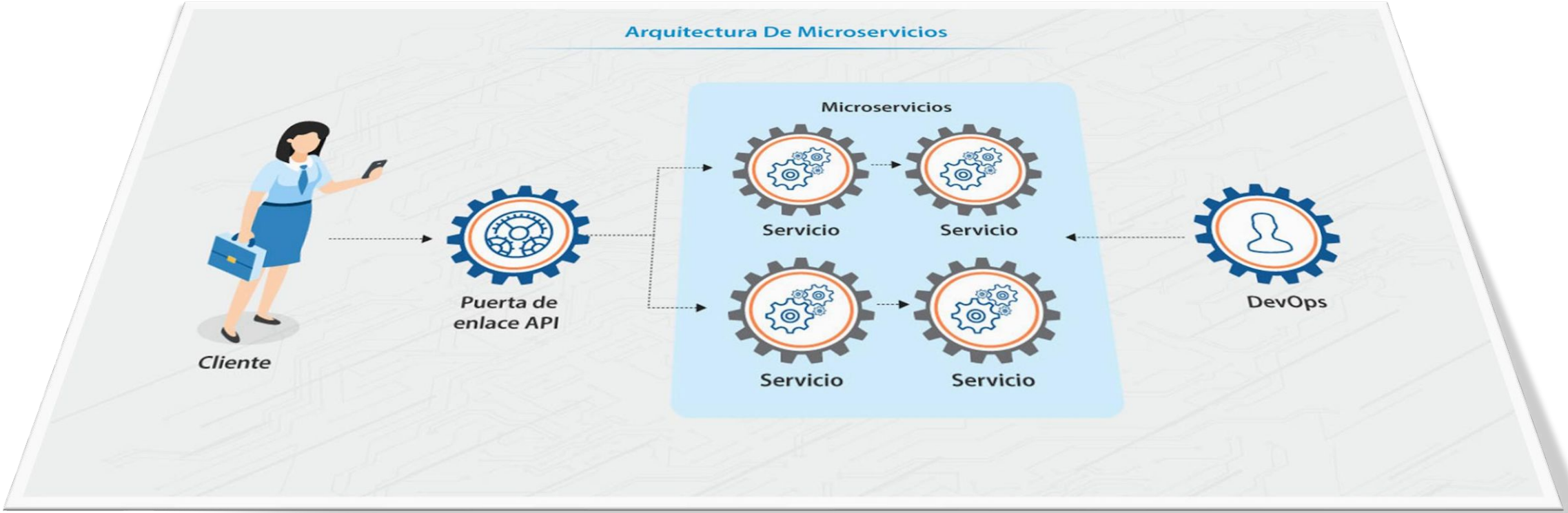
# PRACTICA TRABAJO AUTONOMO CON DOCKER

- **Instale DOCKER en su sistema operativo.**
- **Tome como referencia este <u>documento</u>**
- **Después de instalar las bases de datos, construye el modelo de la página 37 en cada una.**
- **Crear un video que muestre paso a paso la instalación de los sistemas manejadores de base de datos vistos.**
- **Publique el video en youtube**

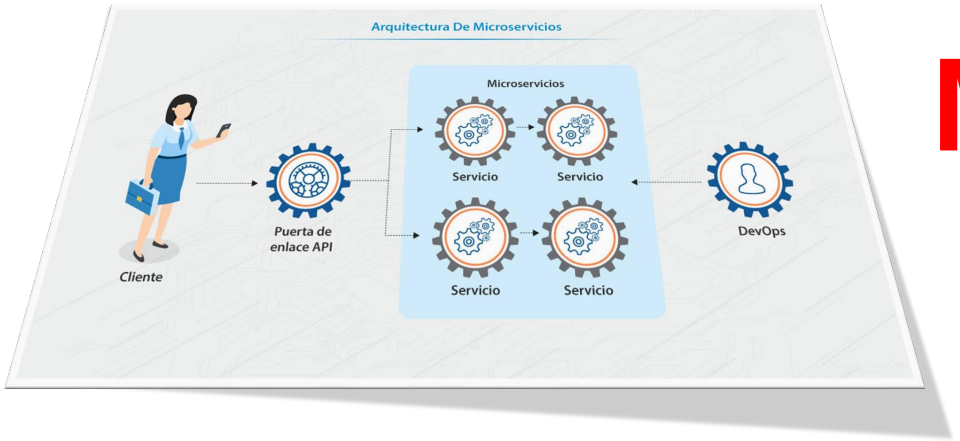
JOSE FERNANDO GALINDO SUAREZ

# Microservicios con APIREST

# MICROSERVICIOS




JOSE FERNANDO GALINDO SUAREZ




# MICROSERVICIOS

**Microservicios**
Son aplicaciones independientes, haciendo una tarea específica (como gestionar usuarios o procesar pagos) dentro de una aplicación más grande.

**Cómo se Relacionan:**

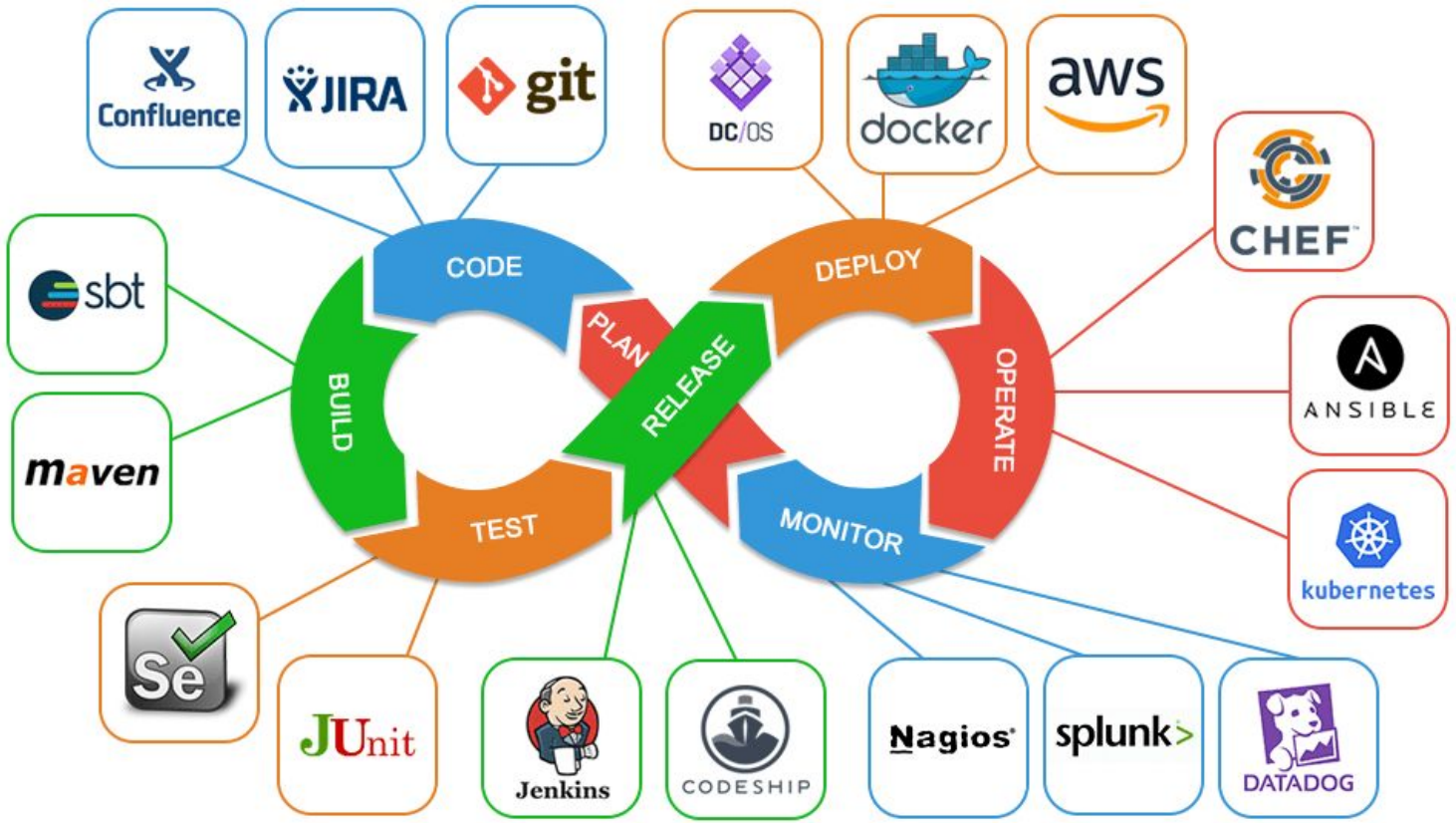
- Los microservicios y las API REST suelen usarse juntos.
- Los microservicios se encargan de dividir una aplicación grande en partes más manejables.
- Las API REST permiten que estas partes se comuniquen y trabajen juntas.
- Esta combinación hace que las aplicaciones sean más flexibles y fáciles de actualizar o expandir.

**API REST**
Conjunto de reglas que permite que estas pequeñas aplicaciones (microservicios) y otros sistemas se comuniquen entre sí a través de Internet.

Tomado de.

JOSE FERNANDO GALINDO SUAREZ

# DESARROLLO Y OPERACIONES



**DESARROLLO Y OPERACIONES**

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

**INTRODUCCIÓN**.
Las API o interfaz de programación de aplicaciones, son un conjunto de funciones que se ofrecen mediante bibliotecas de programación para su uso.
Para entender el concepto de API debemos leer los siguientes términos:

**Interfaz**: capa de extracción para que las aplicaciones se comuniquen.
•**API**: del inglés Application Programming Interface, que permite compartir datos y procesos entre ellos.
•**Arquitectura de software**: Es la forma como está diseñado un sistema, como se comunican al interior, como están organizados sus componentes.
•**Servicio WEB**: Permite una comunicación entre aplicaciones de una red usando el protocolo HTTP.
•**REST**: Es una arquitectura, del inglés Representational State Transfer o Representación de transferencia de estado, permitiendo que datos puede acceder, revisar o manipular otra aplicación.
•**XML**: Formato tradicional para enviar datos, del inglés Extensible Markup Language.
•**JSON**: Formato actual para el intercambio de datos, del inglés JavaScript Object Notation
•**TOKEN: API** pueden ser públicas o privadas, esta última requiere de una autenticación que al hacerlo te devuelve un token que es un objeto que contiene datos de la autenticación y su formato es JWT

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

## Tipos de API

- **Locales**: Se ejecutan dentro del mismo entorno. Por ejemplo cuando se desarrolla en Android y se quiere que el celular vibre, entonces debo utilizar la API del smartphone.
- **Remotas**: Cuando se consumen datos que están en otro lugar

o **Servicios WEB**

- **SOAP** (Simple Object Access Protocol)
- **REST** (Representational state transfer) y al utilizarlo lo llamamos RESTFUL

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

**Cada recurso que se consuma tiene un identificador único (URI); cuando se consulta un recurso el servidor puede contestar con los siguientes códigos:**

- **2XX: Todo fue exitoso**
- **3XX: Significan redirecciones**
- **4XX: Solicitudes invalidas**
- **5XX: Errores directamente en el servidor**

[Fuente]

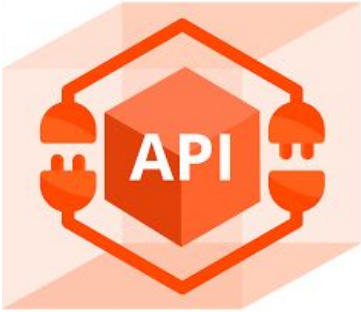
JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

## MÉTODOS HTTP

•**GET**: Este método se emplea para leer una representación de un resource. En caso de respuesta positiva (200 OK), devuelve la representación en un formato concreto: HTML, XML, JSON o imágenes, JavaScript, CSS, etc. Leer mas

•**POST**: Se utiliza POST por las limitaciones de GET. En caso de respuesta positiva devuelve 201 (*created*). Los POST requests se envían normalmente con formularios Leer mas..

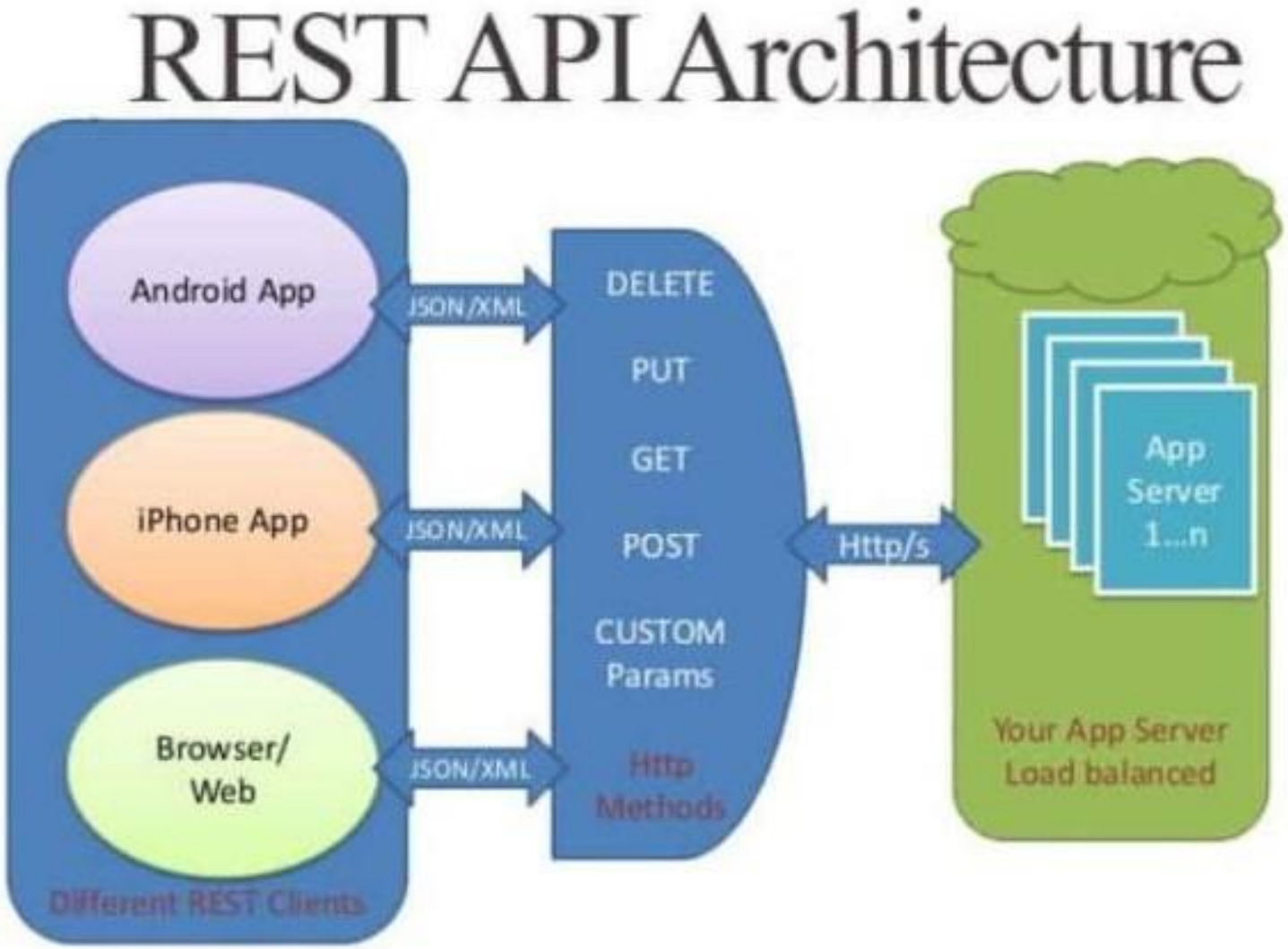



•**PUT**: Utilizado normalmente para actualizar contenidos. Tampoco muestra ninguna información en la URL. En caso de éxito devuelve 201 (created, en caso de que la acción haya creado un elemento) o 204 (*no response*, si el servidor no devuelve ningún contenido). A diferencia de POST es idempotente, (*propiedad para realizar una acción varias veces y aun así conseguir el mismo resultado que se obtendría si se hicieran una sola vez*), si se crea o edita un resource con PUT y se hace el mismo request otra vez, el resource todavía está ahí y mantiene el mismo estado que en la primera llamada. Si con una llamada PUT se cambia aunque sea sólo un contador en el resource, la llamada ya no es idempotente, ya que se cambian contenidos. Leer mas…

•**DELETE**: Simplemente elimina un resource identificado en la URI. Si se elimina correctamente devuelve 200 junto con un body response, o 204 sin body. DELETE, al igual que PUT y GET, también es idempotente, Leer más...

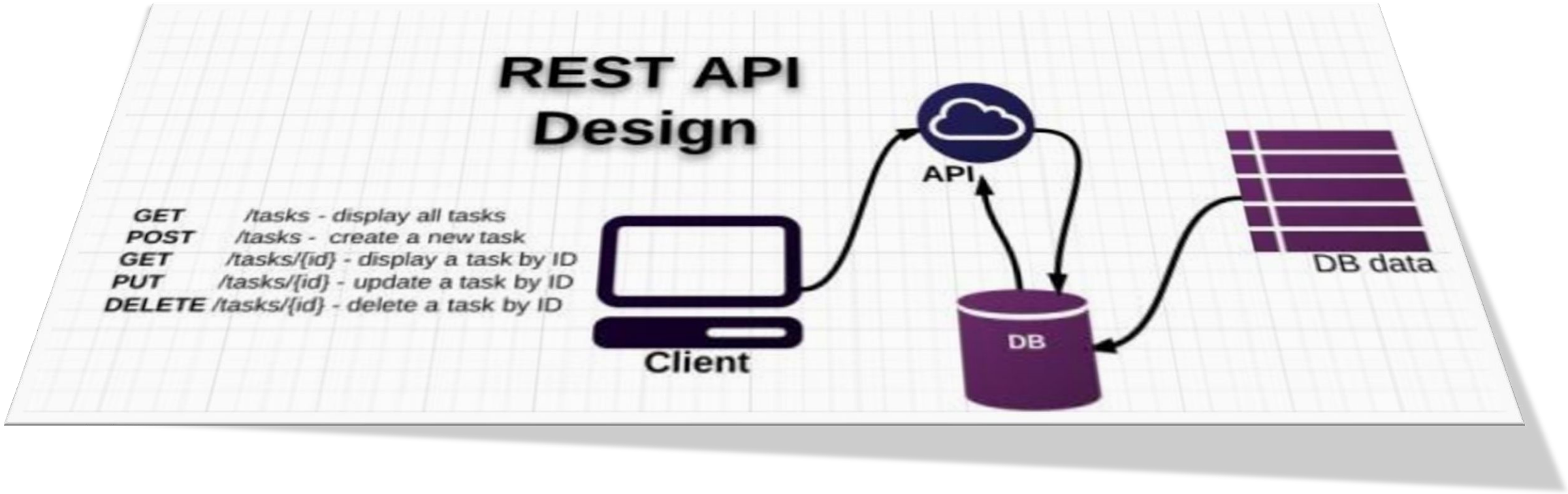
JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API




JOSE FERNANDO GALINDO SUAREZ

# PRACTICA CONECTANDO CON API DESDE PYTHON

**Crear en MYSQL la tabla USUARIO e Insertar los siguientes valores:**

X

```
CREATE DATABASE APIDB;  USE APIDB

CREATE TABLE IF NOT EXISTS USUARIO (
idUsuario INTEGER PRIMARY KEY AUTO_INCREMENT,
login varchar ( 30 ),  nombre TEXT,  apellido TEXT,  email TEXT
);

INSERT INTO USUARIO
(idUsuario,login,nombre,apellido,email)
VALUES (2,'fegasu','JOSE FERNANDO','GALINDO SUAREZ',
'fegasu@misena.edu.co');
INSERT INTO USUARIO
(idUsuario,login,nombre,apellido,email)
VALUES (38,NULL,'Carmen
Julia','Vargas','cvargas@gmail.com');
```

Descargar Talend

JOSE FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL

**PRACTICA DEL TALLER**

**CREANDO LOS MÉTODO DE CONSULTAS DE BASE DE DATOS**

```
from flask import Flask,render_template,request
import json
import mysql.connector
import pandas as pd
```

1

```
conn=mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="APIDB",
    port=3306
)
cursor=conn.cursor()
```

2

```
def Ejecutar(sql):  cursor=cnn.cursor()  cursor.execute(sql)
cnn.commit()
return json.dumps("200")
```

3

```
def EjecutarQuery(sql):  cursor.execute(sql)
row=cursor.fetchall()  return json.dumps(row)
app=Flask(__name__)
```

4

```
if  __name __==“__main __":
app.run(debug=True)
        app.run(debug=True)
```

5

JOSE FERNANDO GALINDO SUAREZ

# TE RETO

# RETO EVIDENCIA

1. **Realizar un video en YouTube que muestre la creación de una APIREST en Python que realice un CRUD en MySQL, POSTGRESQL y MongoDB.**
2. **Se debe realizar desde Python.**
3. **Enviar el script de creación del punto 2.**

**<u>Nota</u>: Esta evidencia se debe realizar en grupo de 4 aprendizaje, se debe escoger un motor de base de datos ( MySQL, POSTGRESQL y MongoDB), y se debe realizar todas las operaciones aprendidas en Python y registrado en el video. Se debe subir individualmente indicando los nombres de sus compañeros de equipo.**

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

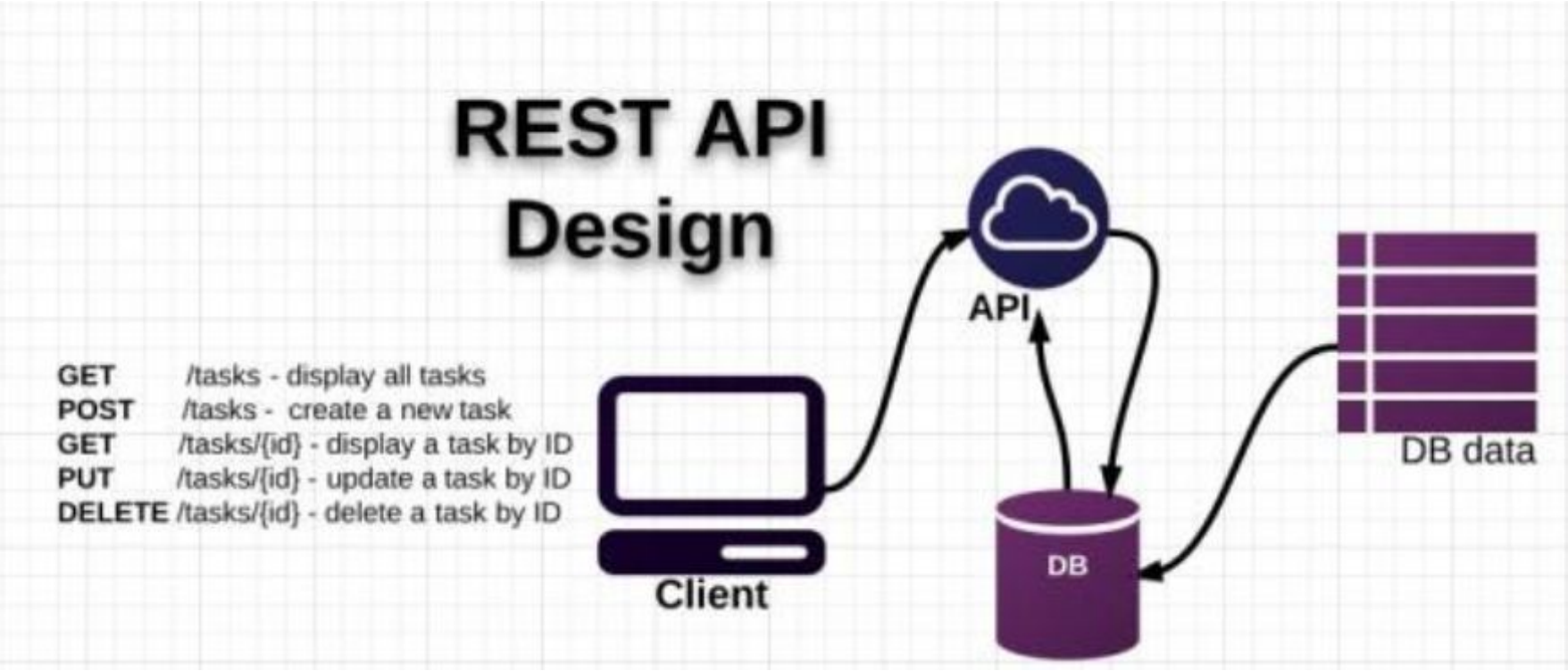
## REST API

| | | |
|---|---|---|
| Crear usuario | POST | /usuarios |
| Obtener un usuario | GET | /usuarios/{id} |
| Obtener usuarios | GET | /usuarios |
| Actualizar usuario | PUT | /usuarios/{id} |
| Eliminar usuario | DELETE | /usuarios/{id} |

**DESCARGAR TALEND APITEST**

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API




JOSE FERNANDO GALINDO SUAREZ

# PRACTICA CONECTANDO UNA API CON PYTHON

# PRACTICA CONECTANDO CON API DESDE PYTHON

**Crear en MYSQL la base de datos APIDB y la tabla USUARIO e Insertar los siguientes valores:**

```
CREATE DATABASE APIDB;
USE APIDB

CREATE TABLE IF NOT EXISTS USUARIO (
idUsuario INTEGER PRIMARY KEY AUTO_INCREMENT,
login varchar ( 30 ),
nombre TEXT,
apellido TEXT,
email TEXT
);

INSERT INTO USUARIO (idUsuario,login,nombre,apellido,email)
VALUES (2,'fegasu','JOSE FERNANDO','GALINDO SUAREZ’,
'fegasu@misena.edu.co');
INSERT INTO USUARIO (idUsuario,login,nombre,apellido,email)
VALUES (38,NULL,'Carmen Julia','Vargas','cvargas@gmail.com');
```

**DESCARGAR API-TEST**

JOSE FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL

```
from flask import Flask,render_template,request
import json
import mysql.connector
import pandas as pd
```

```
conn=mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="",
    database="APIDB",
    port=3306
)
cursor=conn.cursor()
```

```
def Ejecutar(sql):
    cursor=cnn.cursor()
    cursor.execute(sql)
    cnn.commit()
    return json.dumps("200")
```

**PRACTICA DEL TALLER**

**CREANDO LOS MÉTODO DE CONSULTAS DE BASE DE DATOS**

```
if __name__ =="__main__":
    app.run(debug=True)
```

```
def EjecutarQuery(sql):
    cursor.execute(sql)
    row=cursor.fetchall()
    return json.dumps(row)
app=Flask(__name__)
```

DOCENTE: FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL

## CREANDO LA RUTA PARA CONSULTAR CON EL MÉTODO GET

| | | |
|---|---|---|
| Obtener usuarios | GET | /usuarios |
| Obtener un usuario | GET | /usuarios/{id} |

```
@app.route("/usua",methods=['GET'])
def get_usuarios():
   sql="select * from usuario order by 1"
   listar=EjecutarQuery(sql)
   return listar

@app.route("/usua/s/<id>",methods=['GET'])
def get_usuarios1(id):
   sql="select * from usuario where idusuario="+str(id)+" order by 1"
   listar=EjecutarQuery(sql)
   return listar
```

JOSE FERNANDO GALINDO SUAREZ

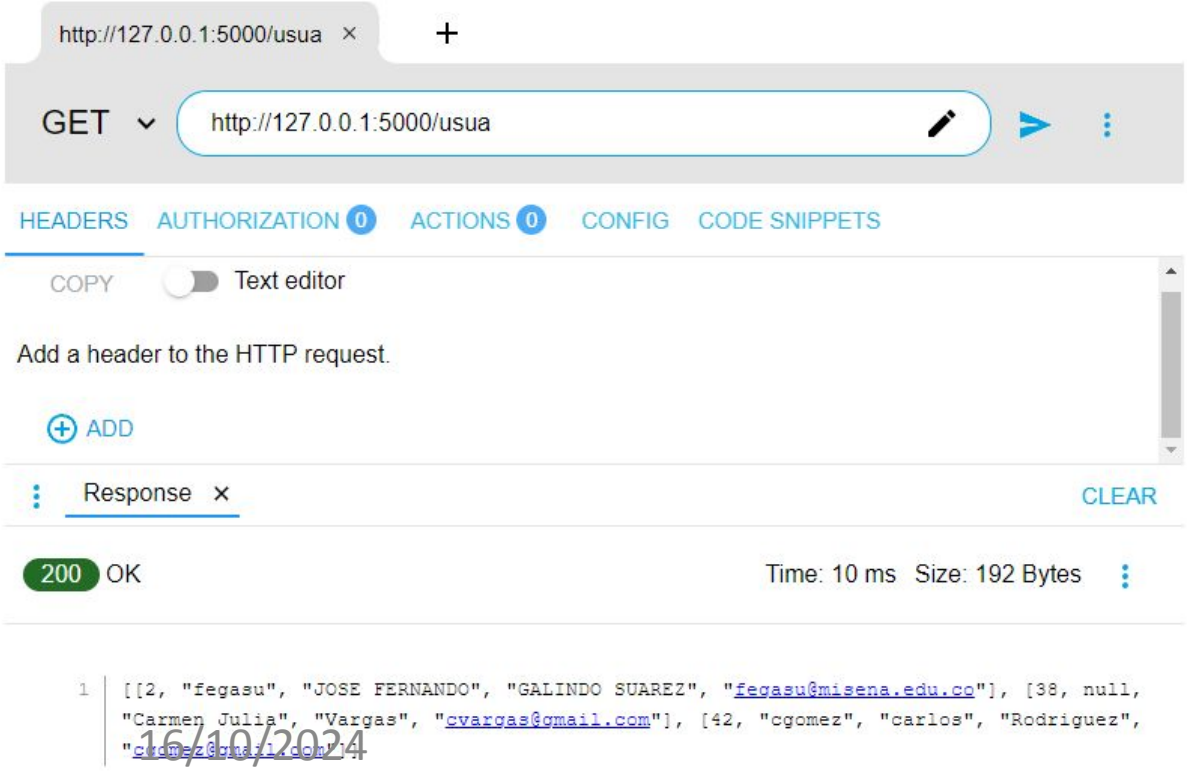
# CONSTRUIR UNA API CON RESTFULL

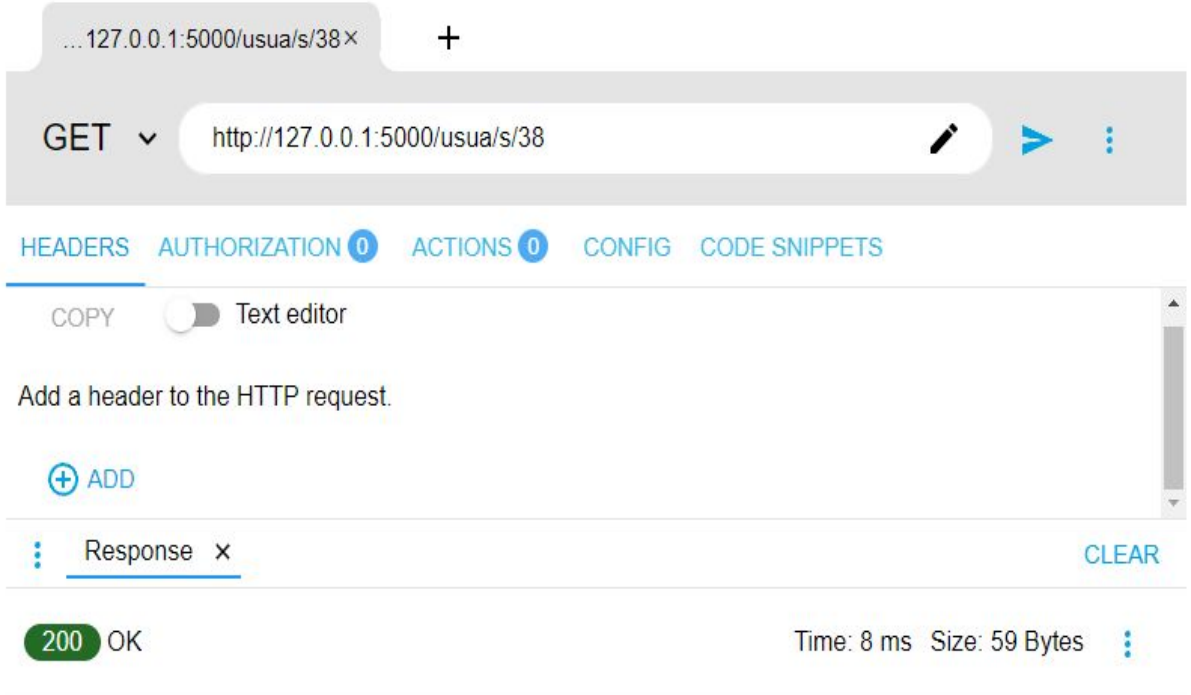
**http://127.0.0.1:5000/usua**

[[2, "fegasu", "JOSE FERNANDO", "GALINDO SUAREZ", "fegasu@misena.edu.co"], [38, null, "Carmen Julia", "Vargas", "cvargas@gmail.com"], [42, "cgomez", "carlos", "Rodriguez", "cgomez@gmail.com"]]

**http://127.0.0.1:5000/usua/s/38**

[[38, null, "Carmen Julia", "Vargas", "cvargas@gmail.com"]]




JOSE FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL

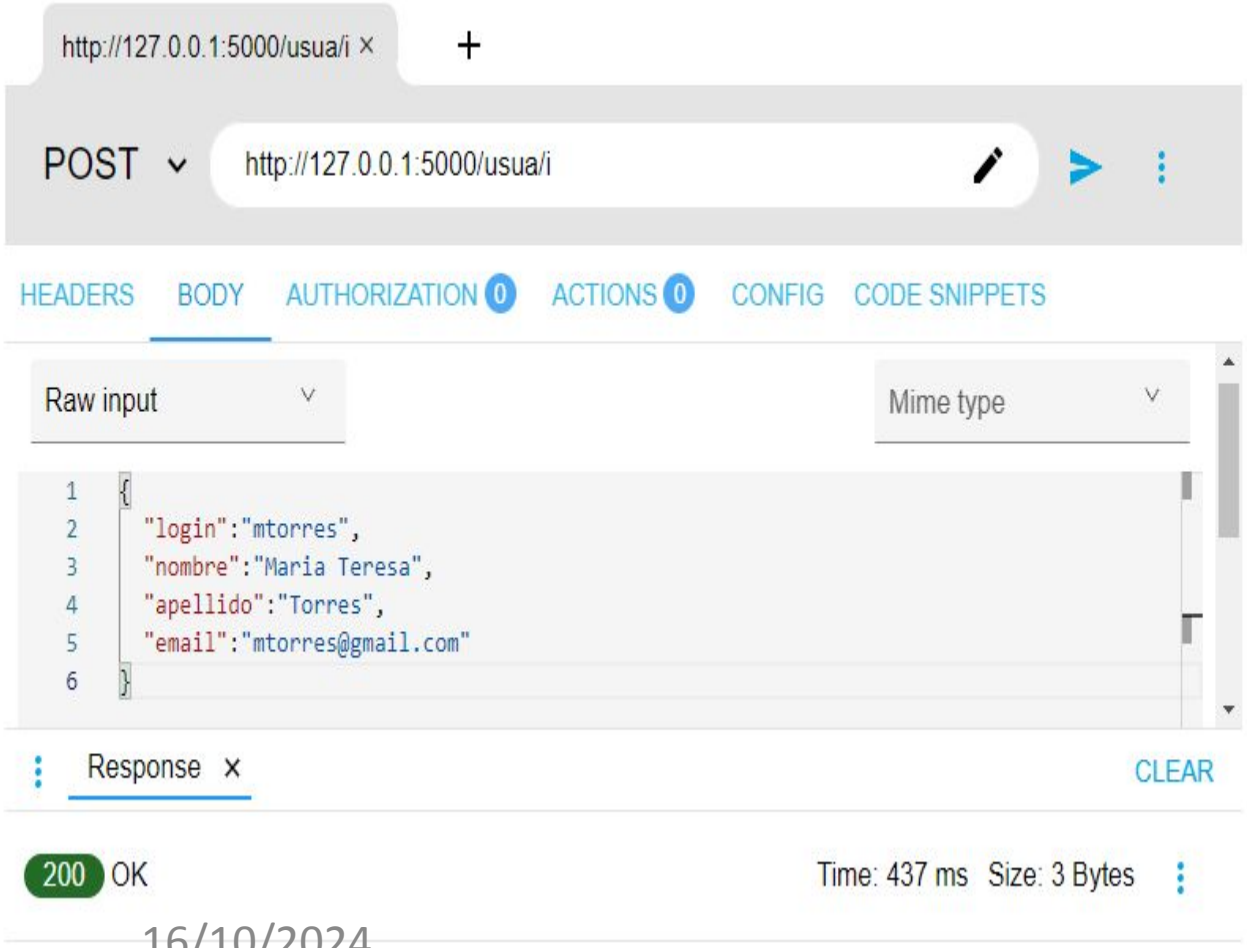
## EJECUTANDO EL SERVICIO POST QUE INSERTA UN REGISTRO EN LA BASE DE DATOS

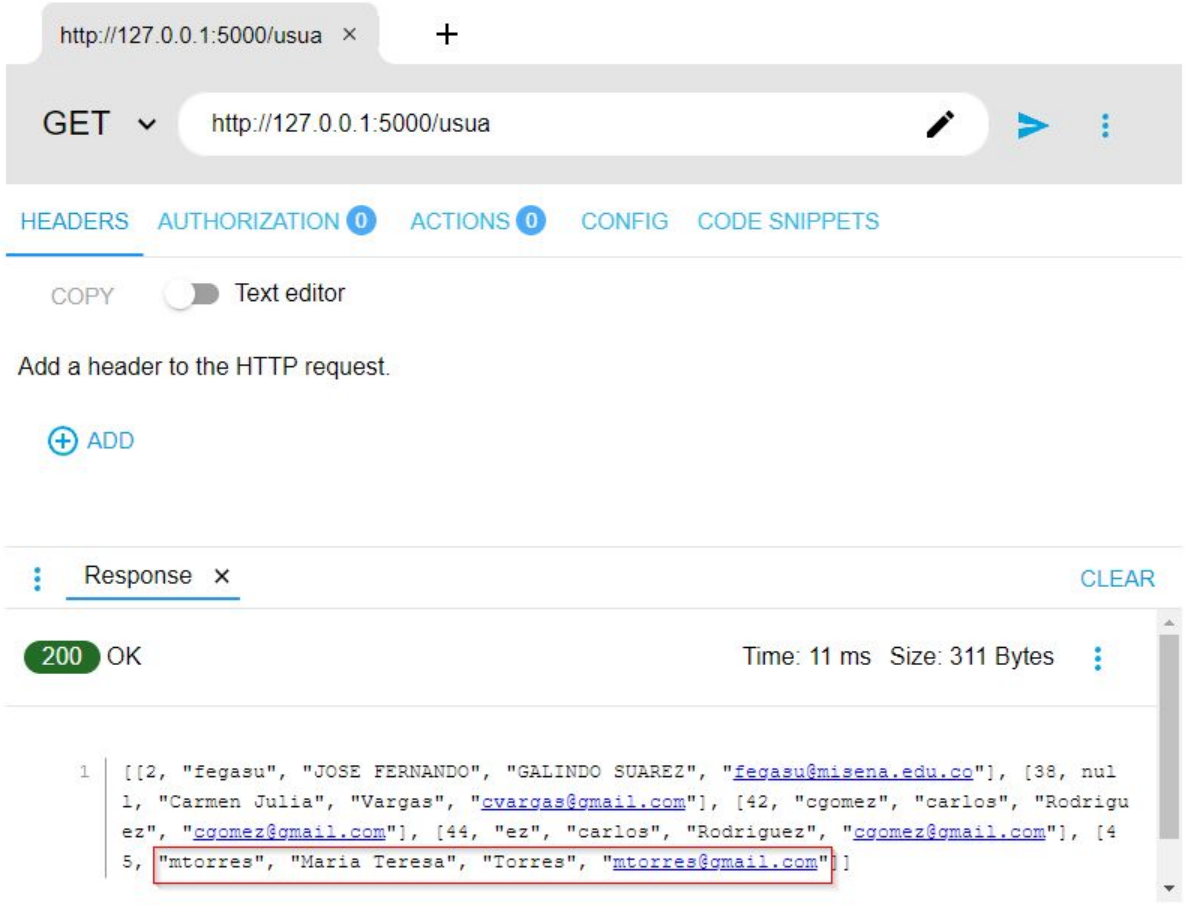
| Crear usuario | POST | /usuarios |
|---|---|---|

```
@app.route("/usua/i",methods=['POST'])
def post_iusuario():
    json=request.get_json(force=True)
    sql="insert into usuario(login,nombre,apellido,email) values(%s,%s,%s,%s)"
    sqlv=(json["login"],json["nombre"],json["apellido"],json["email"])
    cursor.execute(sql,sqlv)
    conn.commit()
    return "200"
```

JOSE FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL



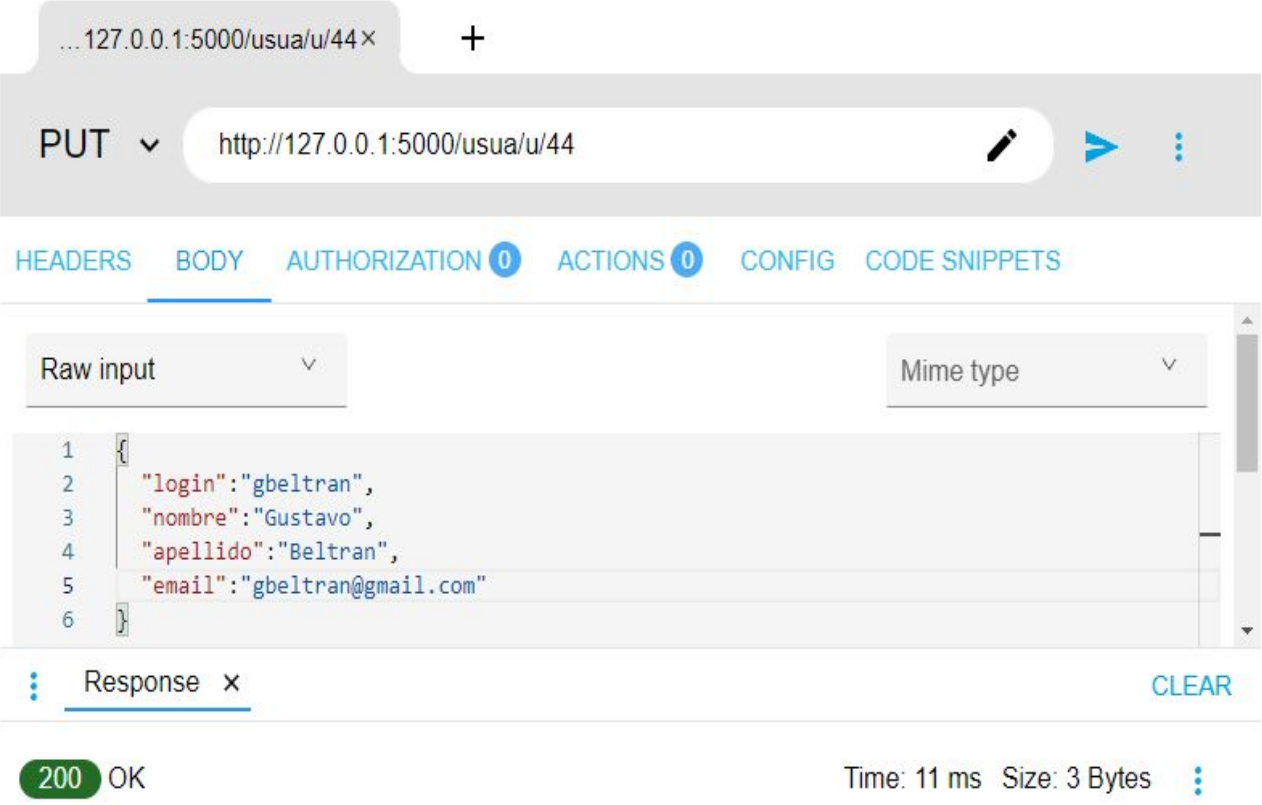
# CONSTRUIR UNA API CON RESTFULL

## EJECUTANDO EL SERVICIO PUT QUE ACTUALIZA UN REGISTRO EN LA BASE DE DATOS

| Actualizar usuario | PUT | /usuarios/{id} |
|---|---|---|

```
@app.route("/usua/u/<id>",methods=['PUT'])
def post_ausuario(id):
    json=request.get_json(force=True)
    sql="update usuario set login=%s,nombre=%s,apellido=%s,email=%s where idusuario="+str(id)
    sqlv=(json["login"],json["nombre"],json["apellido"],json["email"])
    cursor.execute(sql,sqlv)
    conn.commit()
    return "200"
```

# CONSTRUIR UNA API CON RESTFULL



# CONSTRUIR UNA API CON RESTFULL

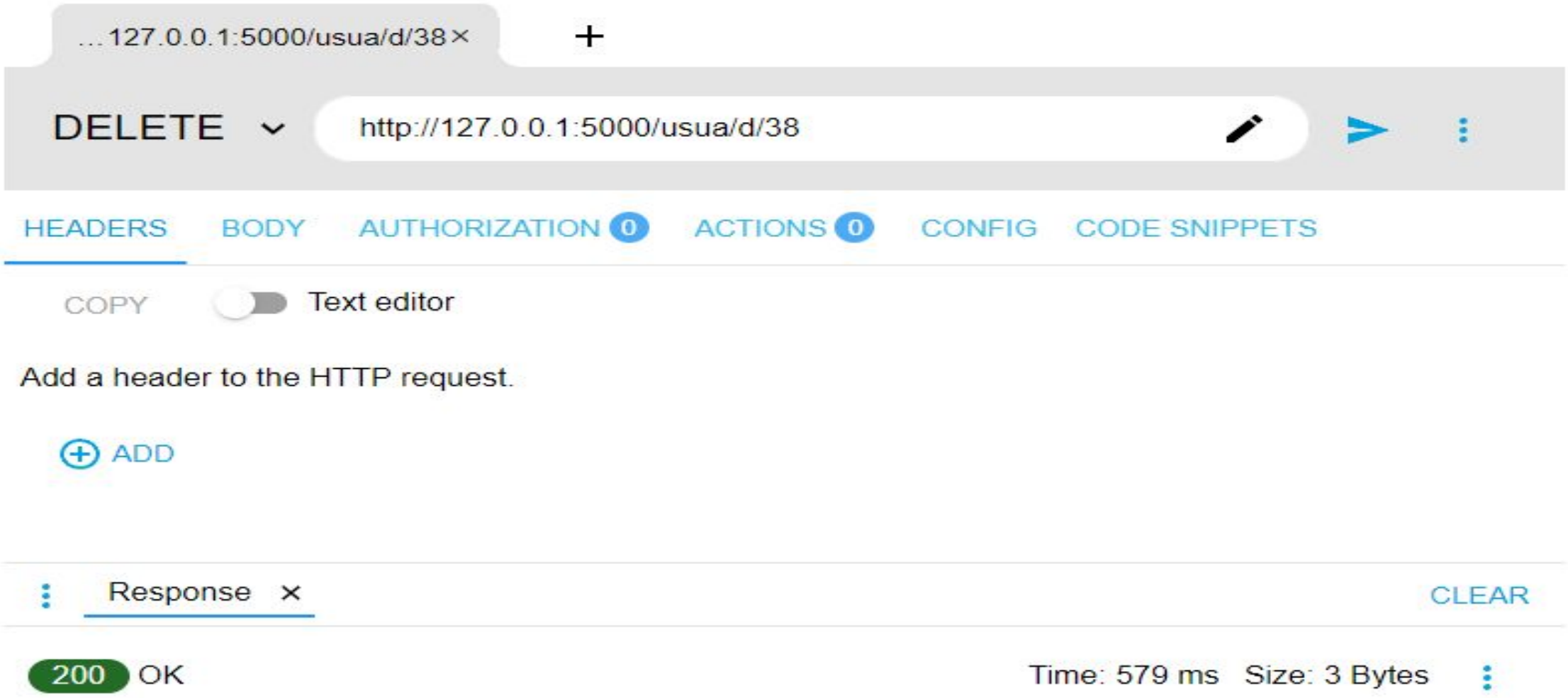
## EJECUTANDO EL SERVICIO DELETE QUE ELIMINA UN REGISTRO EN LA BASE DE DATOS

| Eliminar usuario | DELETE | /usuarios/{id} |
|---|---|---|

```
@app.route("/usua/d/<id>",methods=['DELETE'])
def post_dusuario(id):
    sql="delete from usuario where idusuario="+str(id)
    cursor.execute(sql)
    conn.commit()
    return "200"
```

JOSE FERNANDO GALINDO SUAREZ

# CONSTRUIR UNA API CON RESTFULL




JOSE FERNANDO GALINDO SUAREZ

# INICIAR SERVIDOR FLASK

**INICIAR EL SERVIDOR DE LA APIREST (start_flask.bat)**

```
@echo off
cls
echo INICIANDO EL SERVIDOR WEB FLASK
set FLASK_APP=index.py
set FLASK_ENV=development
@call flask run
```

Símbolo del sistema - start_flask

```
INICIANDO EL SERVIDOR WEB FLASK
 * Serving Flask app 'index.py'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
```

JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API

## CREAR UN MÉTODO PARA EJECUTAR POR TIPO DE SERVICIO

```
import urllib.request
import json
import requests
from requests import Session

def EjecutaRest(url,metodo,datos):
   url=url.replace(chr(32),'+')
   miurl=None
   try:
      if metodo=="POST":
         miurl=requests.post(url,json=datos)
      if metodo=="PUT":
         miurl=requests.put(url,json=datos)
      if metodo=="DELETE":
         miurl=requests.delete(url)
return miurl
   except ValueError:
      messagebox.showerror("Rutinas/EjecutaRest","Ocurrio un error")
```

JOSE FERNANDO GALINDO SUAREZ

# CONSUMIR LOS SERVICIOS DE LA API

```
datos={
        "login":"mlb",
        "nombre":"Maria",
        "apellido":"La bandida",
        "email":"emadera@gmail.com"
}
```

```
#print(EjecutaRest('http://127.0.0.1:5000/usua/i','POST',datos))
#print(EjecutaRest('http://127.0.0.1:5000/usua/u/3','PUT',datos))
#print(EjecutaRest('http://127.0.0.1:5000/usua/d/3','DELETE',datos))
```

JOSE FERNANDO GALINDO SUAREZ

# EJEMPLO CONSUMIR LOS SERVICIOS GET DE LA API

**Crear un archivo llamado rutinas.py**

```
import mysql.connector  import urllib.request
import json
from tkinter import *  from tkinter import ttk
```

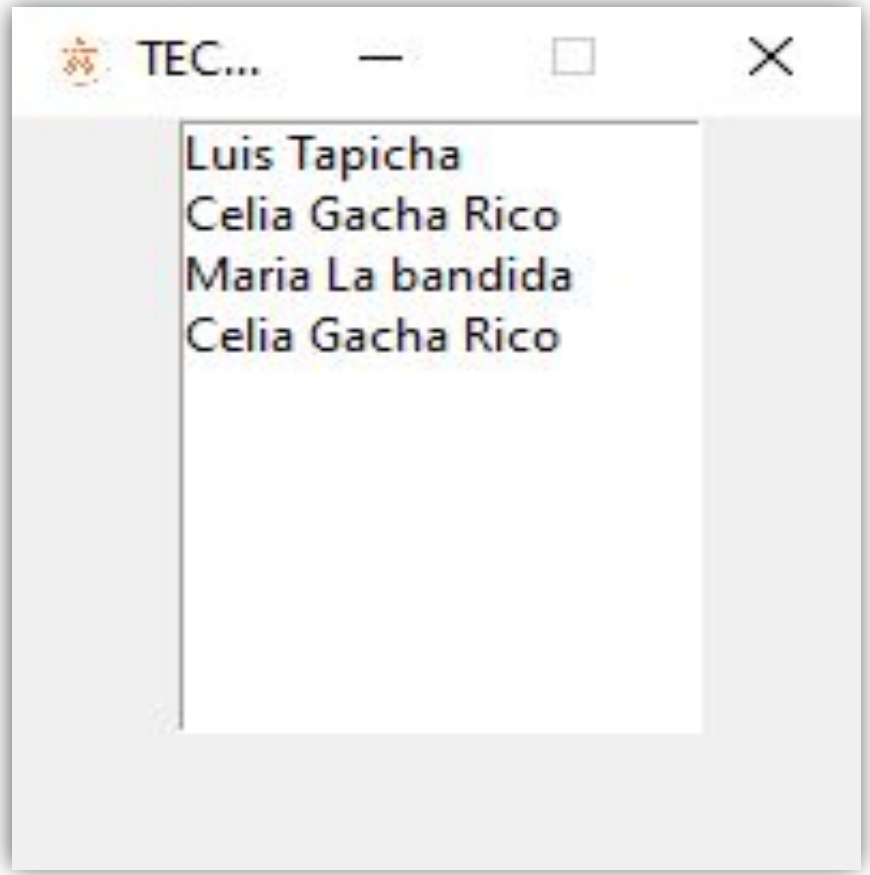
```
def CargaDato(url):  url=url.replace(chr(32),'+')
miurl=urllib.request.urlopen(url)
miarreglo=json.loads(miurl.read().strip())
miarreglo.reverse()
return miarreglo
```

```
class kinter:
def Ventana(self,titu,tam):  v=
Tk();v.title(titu);v.resizable(0,0);v.geometry(tam)  return v
def AgregarLista(self,Datos,lista):
for i in range(len(Datos)):  lista.insert(Datos[i][0],Datos[i][2]+"
"+Datos[i][3])
lista.pack()
def CrearLista(self,v):
nom=Listbox(v);nom.grid(column=0,row=0)  return nom
```

APIS PUBLICAS

# EJEMPLO CONSUMIR LOS SERVICIOS GET DE LA API

```
# Crear un archivo llamado consultaapi.py

import pandas as pd
from rutinas import *

Datos=CargaDato('http://127.0.0.1:5000/usua')
Datos1=pd.DataFrame(CargaDato('http://127.0.0.1:5000/usua/c/usuario'))
p1=kinter()
v0=p1.Ventana('TECNOLOGIAS EMERGENTES CON PYTHON V1.0 2023','200x200’)
# v0.iconbitmap("img/logosenacir.ico")
lista2=p1.CrearLista(v0)
p1.AgregarLista(Datos,lista2)
v0.mainloop()
```




JOSE FERNANDO GALINDO SUAREZ

# CONECTANDO CON API EXTERNA

## EJEMPLO CONSUMIR LOS SERVICIOS GET DE LA API

Crear un archivo llam

```
import pandas as pd
import mysql.connector
import urllib.request
import json
from tkinter import *
from tkinter import ttk
```

```
def CargaDato(url):
    url=url.replace(chr(32),'+')
    miurl=urllib.request.urlopen(url)
    miarreglo=json.loads(miurl.read().strip())
    miarreglo.reverse()
    return miarreglo
```

```
class kinter:
    def Ventana(self,titu,tam):
        self.v=Tk()
        self.v.title(titu)
        self.v.resizable(0,0)
        self.v.geometry(tam)
        return self.v
    def AgregarLista(self,Datos,lista):
        for i in range(len(Datos)):

aux="["+Datos[i]['startPeriodDate']+"]-["+Datos[i]['lastNa
me']+" "+Datos[i]['name']+"]"
            lista.insert(i,aux)
            lista.pack()
    def CrearLista(self,v):
        nom=Listbox(v,width=60)
        nom.grid(column=0,row=0)
        return nom
```

Tkinter

APIS PUBLICAS

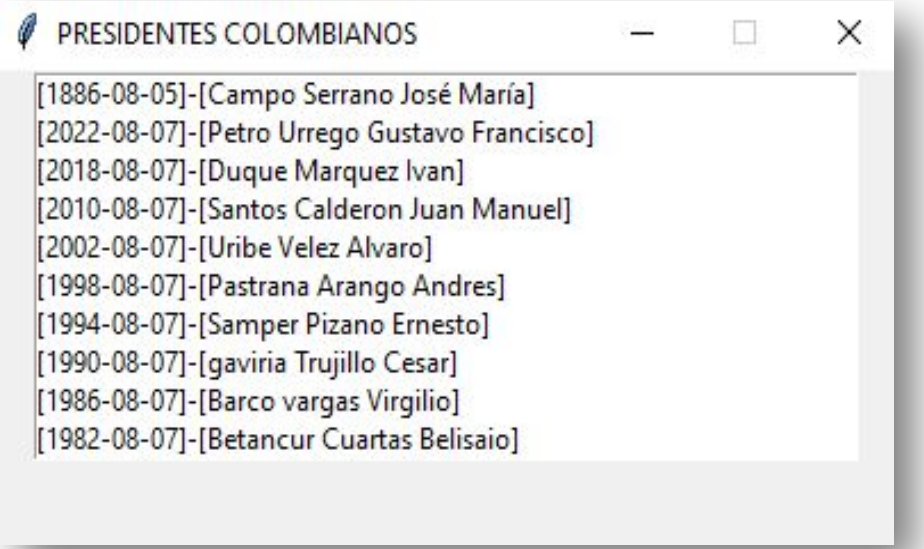
PRESIDENTES COLOMBIA



JOSE FERNANDO GALINDO SUAREZ

# EJEMPLO CONSUMIR LOS SERVICIOS GET DE LA API

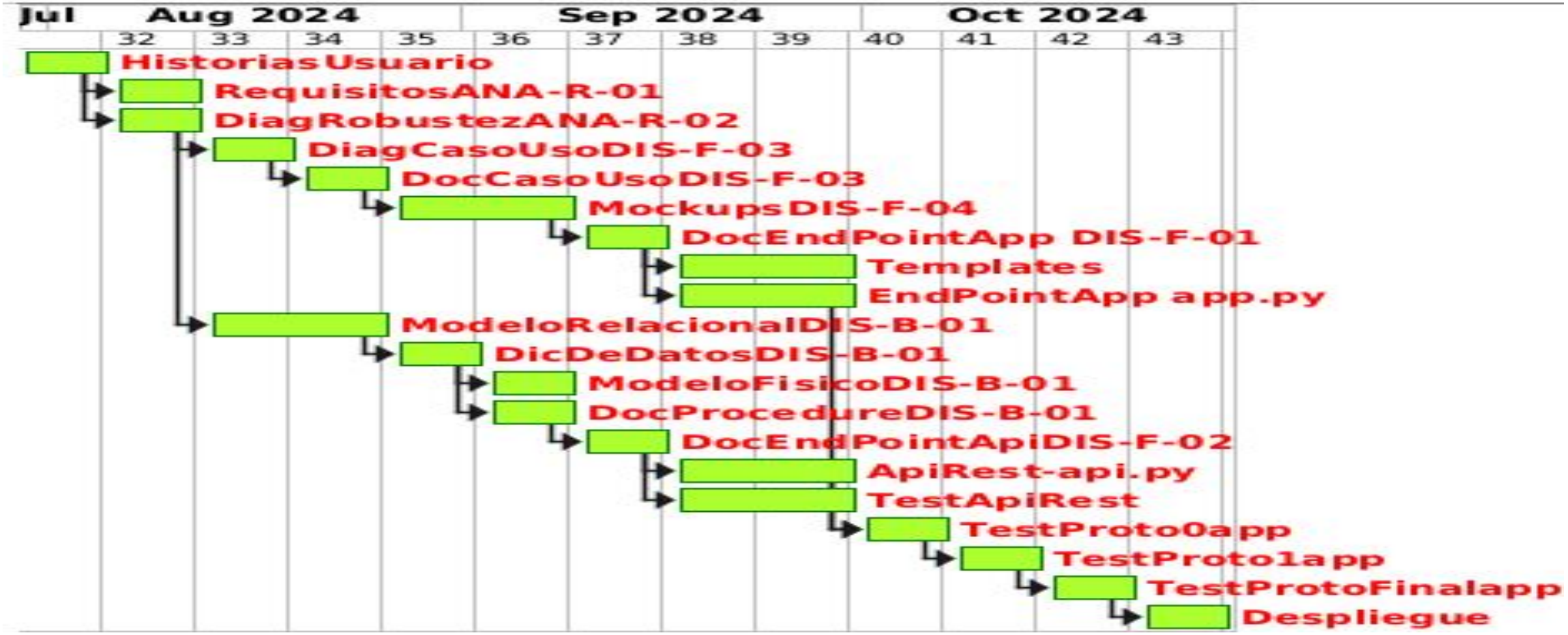
```
# Crear un archivo llamado consultaapi.py
Datos=CargaDato('https://api-colombia.com/api/v1/President')
p1=kinter()
v0=p1.Ventana('PRESIDENTES COLOMBIANOS','400x200')
lista2=p1.CrearLista(v0)
p1.AgregarLista(Datos,lista2)
v0.mainloop()
```




Tkinter

JOSE FERNANDO GALINDO SUAREZ

# ACTIVIDADES APLICADAS EN LA FABRICA DE SOFTWARE




JOSE FERNANDO GALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

Los 5 principios SOLID de diseño de aplicaciones de software son:

- S – Single Responsibility Principle (SRP)
- O – Open/Closed Principle (OCP)
- L – Liskov Substitution Principle (LSP)
- I – Interface Segregation Principle (ISP)
- D – Dependency Inversion Principle (DIP)

JOSE FERNANDO GALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

## Acoplamiento

**El acoplamiento se refiere al grado de interdependencia que tienen dos unidades de software entre sí, entendiendo por unidades de software: clases, subtipos, métodos, módulos, funciones, bibliotecas, etc.**
**Si dos unidades de software son completamente independientes la una de la otra, decimos que están desacopladas.**

## Cohesión

**La cohesión de software es el grado en que elementos diferentes de un sistema permanecen unidos para alcanzar un mejor resultado que si trabajaran por separado.**
**Se refiere a la forma en que podemos agrupar diversas unidades de software para crear una unidad mayor.**

https://profile.es/blog/principios-solid-desarrollo-software-calidad/

JOSE FERNANDO GALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

La "S" del acrónimo "**Single Responsibility Principle (SRP)"**, que establece que “una clase debería tener una, y solo una, razón para cambiar”, Robert C. Martin lo define como “responsabilidad”. Tambien se define como: “Reúne las cosas que cambian por las mismas razones y separa aquellas que cambian por razones diferentes”.

https://profile.es/blog/principios-solid-desarrollo-software-calidad/

JOSE FERNANDO CALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

**Principio de Abierto/Cerrado**

Bertrand Meyer en 1988 en su libro "Object Oriented Software Construction", escribe: "Deberías ser capaz de extender el comportamiento de una clase, sin modificarla". Donde, las clases que usas deberían estar abiertas para poder extenderse y cerradas para modificarse.

Robert C. Martin defendió este principio que a priori puede parecer una paradoja. Es importante tener en cuenta el Open/Closed Principle (OCP) a la hora de desarrollar clases, librerías o frameworks.

https://profile.es/blog/principios-solid-desarrollo-software-calidad/

JOSE FERNANDO GALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

**Principio de Sustitución de Liskov**

La L de SOLID alude al apellido de quien lo creó, Barbara Liskov, y dice que “las clases derivadas deben poder sustituirse por sus clases base”.

Esto significa que los objetos deben poder ser reemplazados por instancias de sus subtipos sin alterar el correcto funcionamiento del sistema o lo que es lo mismo: si en un programa utilizamos cierta clase, deberíamos poder usar cualquiera de sus subclases sin interferir en la funcionalidad del programa.

Según Robert C. Martin incumplir el Liskov Substitution Principle (LSP) implica violar también el principio de Abierto/Cerrado.

JOSE FERNANDO GALINDO SUAREZ



https://profile.es/blog/principios-solid-desarrollo-software-calidad/

# DISEÑO DE SOFTWARE SOLID

**Principio de Segregación de la Interfaz**

"Haz interfaces que sean específicas para un tipo de cliente", es decir, para una finalidad concreta.

En este sentido, según el Interface Segregation Principle (ISP), es preferible contar con muchas interfaces que definan pocos métodos que tener una interface forzada a implementar muchos métodos a los que no dará uso.

JOSE FERNANDO GALINDO SUAREZ

# DISEÑO DE SOFTWARE SOLID

**Principio de Inversión de Dependencias**

"Depend on abstractions, not on concretions.", traducido "Depende de abstracciones, no de clases concretas".

Robert C. Martin recomienda:

- Los módulos de alto nivel no deberían depender de módulos de bajo nivel. Ambos deberían depender de abstracciones.
- Las abstracciones no deberían depender de los detalles. Los detalles deberían depender de las abstracciones.

El objetivo del Dependency Inversion Principle (DIP) consiste en reducir las dependencias entre los módulos del código, es decir, alcanzar un bajo acoplamiento de las clases.

JOSE FERNANDO GALINDO SUAREZ



https://profile.es/blog/principios-solid-desarrollo-software-calidad/

# DUDAS Y PREGUNTAS

JOSE FERNANDO GALINDO SUAREZ

# MATERIALES DE FORMACIÓN

Centro de Gestión de Mercados, Logística y Tecnologías De la Información

Regional Distrito Capital

DATASET NEPTUNO

EXCEL NEPTUNO

JOSE FERNANDO GALINDO SUAREZ

# CUENTAME CÓMO TE FUE?

JOSE FERNANDO GALINDO SUAREZ

# CREDITOS

Atribución, no comercial,compartir igual
Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.
SENA
Creative Commons

# G R A C I A S

Línea de atención al ciudadano: 01 8000 910270
Línea de atención al empresario: 01 8000 910682

@SENAComunica

www.sena.edu.co



https://cienciadedatos.net/machine-learning-python.html