

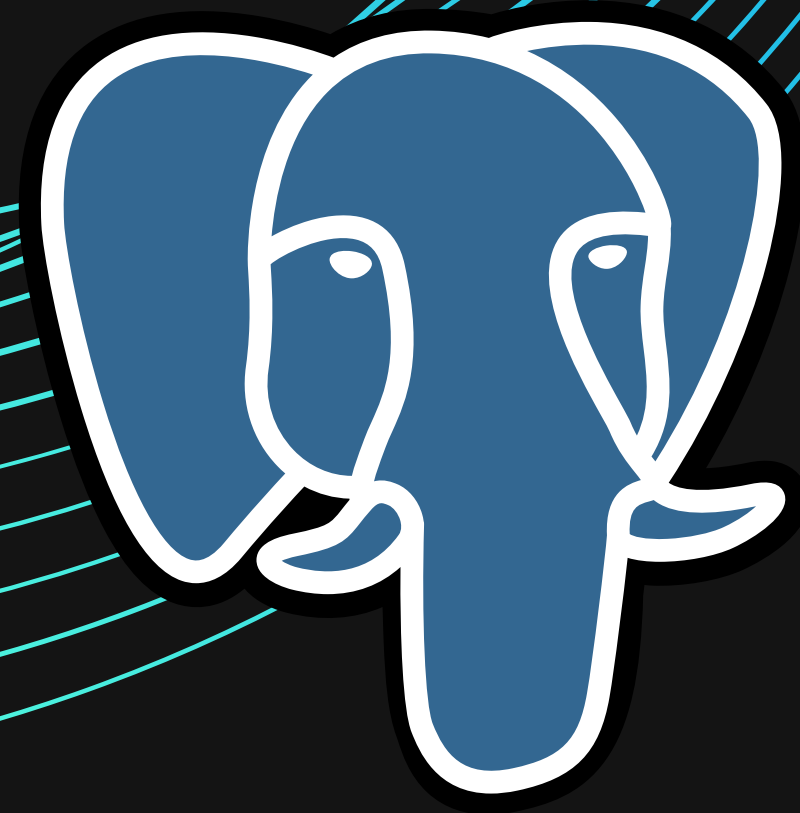
Taller Diseño

WEB CON

DJANGO

Clase #4 : Bases

de datos.




heroku



mongoDB



Contenido



EXPLICACIÓN GENERAL DE UNA BASE DE DATOS

EXPLICACIÓN DE DB EN DJANGO

TABLAS DE DATOS RELACIONALES

Bases de datos

Programa capaz de almacenar gran cantidad de datos, relacionados y estructurados, que pueden ser consultados rápidamente de acuerdo con las características selectivas que se deseen.

Una base de datos es una recopilación organizada de información o datos estructurados, que normalmente se almacena de forma electrónica en un sistema informático. La mayoría de las bases de datos utilizan un lenguaje de consulta estructurada (SQL) para escribir y consultar datos.



Bases de datos en Django

Esta es tal vez la característica más atractiva que tiene Django a nivel profesional.

No hay que saber literalmente nada acerca de bases de datos para poder crear una aplicación web totalmente funcional, ya que Django se encarga 100% de crearla, y administrarla.

Tal vez el único conocimiento base que habría que manejar o al menos conocer es saber hacer clases en python, pero no es nada que no se pueda aprender de manera sencilla.



Python Class
(Inherits From Django Models)

Class Product:
id = integer
name = string
price = float



Database Table

ID	Name	Price
1	Chair	\$12.99
2	Table	\$215.00
3	Ball	\$50.00
4	Couch	\$498.00
5	Pool	\$1600.00

```
class Customer(models.Model):  
    name = models.CharField(max_length=200, null=True)  
    phone = models.CharField(max_length=200, null=True)  
    email = models.CharField(max_length=200, null=True)  
    date_created = models.DateTimeField(auto_now_add=True, null=True)  
  
    def __str__(self):  
        return self.name
```

```
class Customer(models.Model):  
    name = models.CharField(max_length=200, null=True)  
    phone = models.CharField(max_length=200, null=True)  
    email = models.CharField(max_length=200, null=True)  
    date_created = models.DateTimeField(auto_now_add=True, null=True)  
  
    def __str__(self):  
        return self.name
```

Local Library

Site administration | Django si...

+

127.0.0.1:8000/admin/

Search

Disable

Cookies

CSS

Forms

Images

Information

Miscellaneous

Outline

Resize

Tools

View Source

Django administration

WELCOME, UBUNTU. [VIEW SITE](#) / [CHANGE PASSWORD](#) / [LOG OUT](#)

Site administration

AUTHENTICATION AND AUTHORIZATION

Groups

+ Add

Change

Users

+ Add

Change

CATALOG

Authors

+ Add

Change

Book instances

+ Add

Change

Books

+ Add

Change

Genres

+ Add

Change

Recent actions

My actions

None available

Bases de datos relacionales

Una *base de datos relacional* es un tipo de base de datos que almacena y proporciona acceso a puntos de datos relacionados entre sí. Las bases de datos relacionales se basan en el modelo relacional, una forma intuitiva y directa de representar datos en tablas. En una base de datos relacional, cada fila en una tabla es un registro con una ID única, llamada *clave*. Las columnas de la tabla contienen los atributos de los datos y cada registro suele tener un valor para cada atributo, lo que simplifica la creación de relaciones entre los puntos de datos.



PostgreSQL



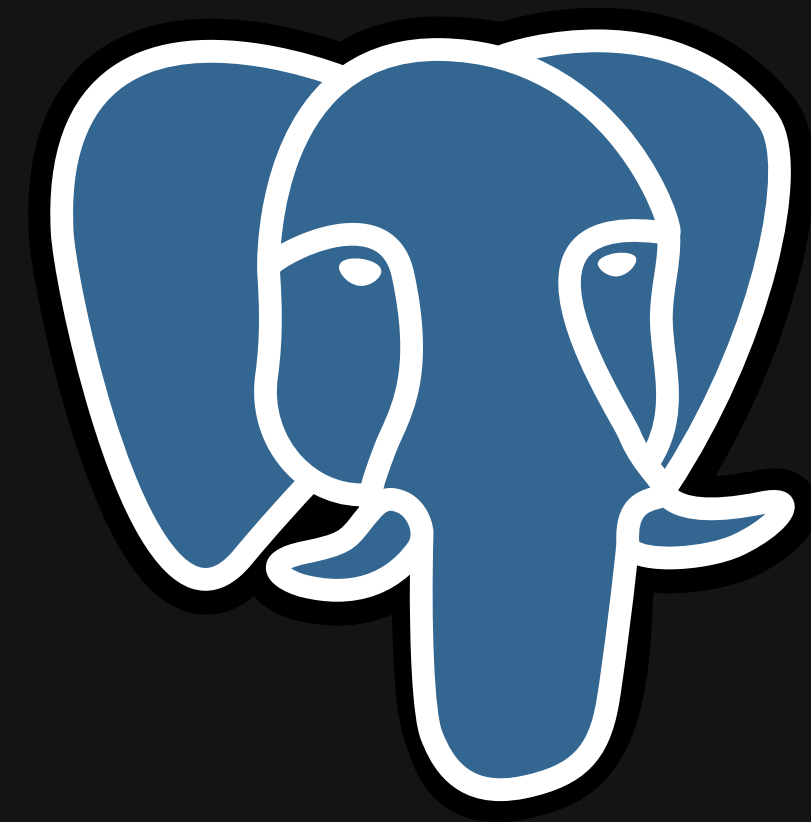
mongoDB

Bases de datos relacionales

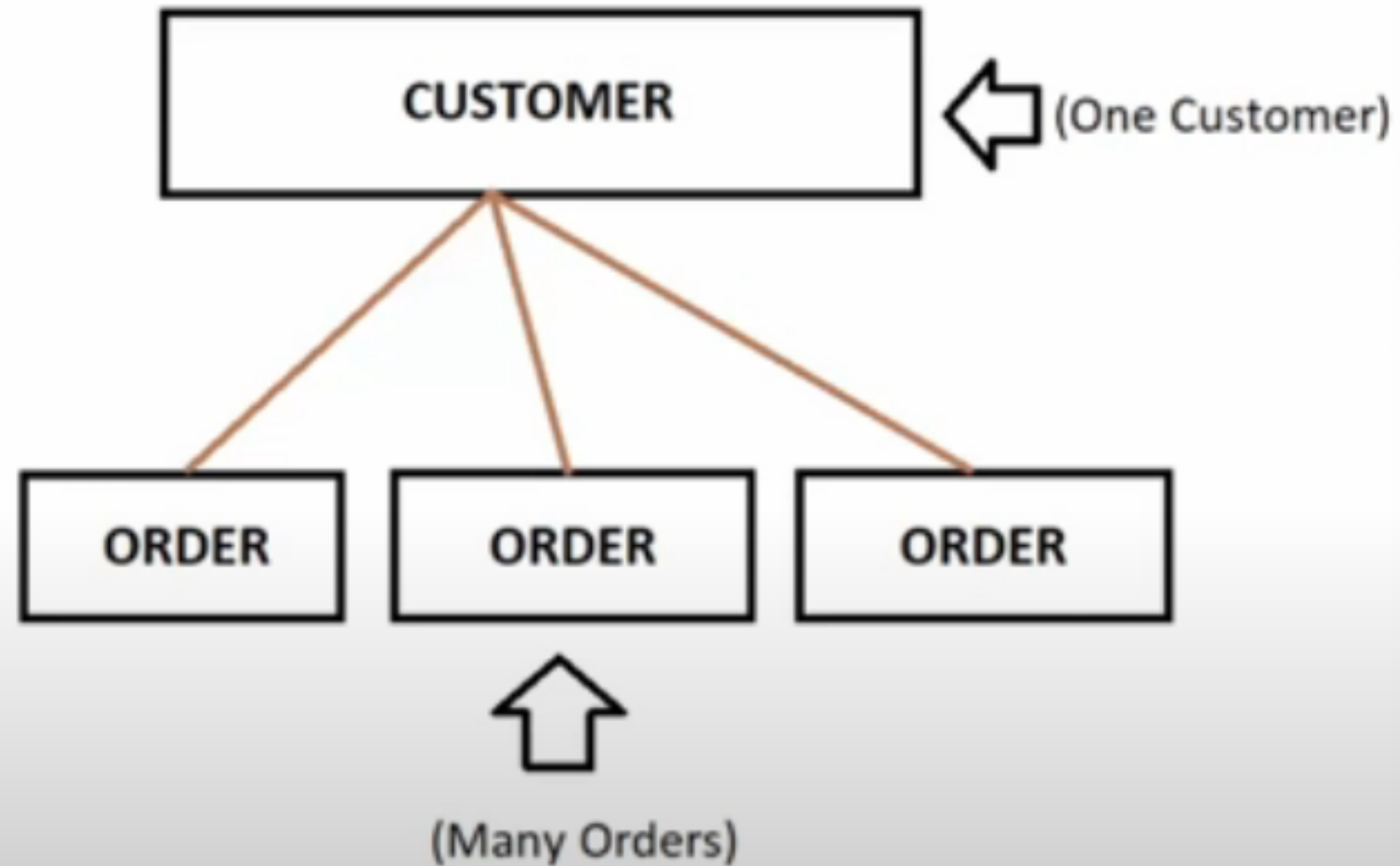
Relacion Uno a Uno: Una Relación uno a uno es un vínculo entre la información de dos tablas, donde cada registro en cada tabla solo aparece una vez. Por ejemplo, puede haber una relación uno a uno entre los empleados y los coches que conducen.

Relación Uno a Muchos: En una relación de uno a muchos, un registro de una tabla se puede asociar a uno o varios registros de otra tabla.. Por ejemplo, cada cliente puede tener varios pedidos de ventas.

Relación Muchos a Muchos: Una relación de muchos a muchos se produce cuando varios registros de una tabla se asocian a varios registros de otra tabla.



One (customer) to Many (Orders)



(Customers)

ID	NAME
1	John
2	Tim
3	Erik

(ORDERS)

ID	CUS_ID	PRODUCT
1	1	Basketball
2	1	BBQ Grill
3	3	Dishset
4	2	Basketball
5	2	Couch
6	3	BBQ Grill

The diagram shows the relationship between the two tables. Lines connect the IDs in the Customers table to the CUS_ID values in the Orders table: John (ID 1) connects to Basketball (ID 1) and BBQ Grill (ID 2); Tim (ID 2) connects to Basketball (ID 4) and Couch (ID 5); Erik (ID 3) connects to Dishset (ID 3) and BBQ Grill (ID 6).

Tags to Product: Many to Many ✓



Table Structure

(Tags)

ID	NAME
1	Sports
2	Summer
3	Kitchen

(Intermediary Table)

tag_id	prdt_id
1	1
2	1
3	2
3	3
2	2

(Products)

ID	PRODUCT
1	Ball
2	BBQ Grill
3	Dishes