EXERCISES QUE TRABAJAREMOS EN EL CUE

- **EXERCISE 1: ACCESO A DATOS CON DJANGO**
- **EXERCISE 2: PROYECTO VS. APLICACIÓN**
- **EXERCISE 3: EJEMPLO DE PROYECTO Y APLICACIONES**
- **EXERCISE 4: EJEMPLO DEL PRINCIPIO DE LA REUTILIZACIÓN**
- **EXERCISE 5: EJEMPLO DE APLICACIÓN EN DJANGO EN PROYECTO**
- **EXERCISE 6: CREACIÓN DE LAS TABLAS EN UNA BASE DE DATOS**

EXERCISE 1: ACCESO A DATOS CON DJANGO

Para el acceso de datos en Django, debemos comenzar por el Modelo. Conocemos que Django hace uso del patrón MTV. En este caso conocemos los Templates, las Vistas y los Modelos, que son la base fundamental.

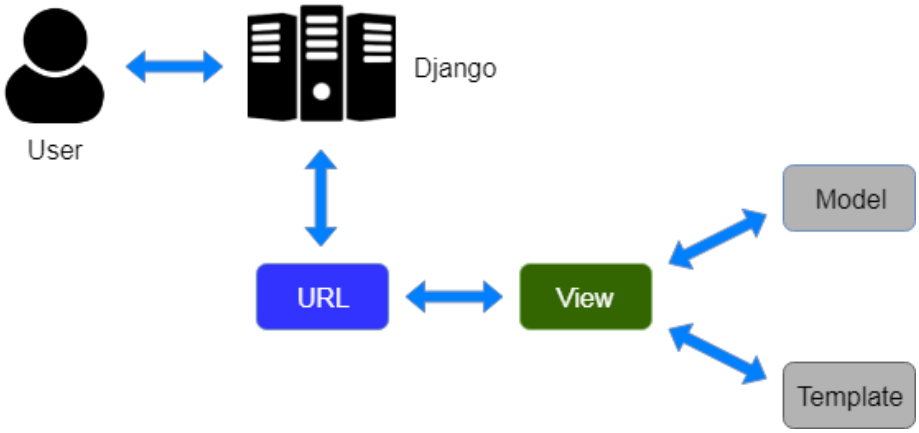

Figura 1.

Recordemos que Django soporta oficialmente la siguiente base de Datos:

- SQLite3 (Gestor de BD por defecto)

  Django permite trabajar por defecto, lo que hace extremadamente fácil utilizar estas bases de datos, ya que no debemos crearlas ni gestionarlas.

- PostgreSQL (Gestor recomendado)
- MySQL
- Oracle

Posee conectores ofrecidos por terceros, tales como:

- SQL Server
- SAP SQL
- DB2
- Firebird

Entre otros.

EXERCISE 2: PROYECTO VS. APLICACIÓN

Django realiza una distinción entre los dos conceptos. Para trabajar con bases de datos se debe tener en cuenta la diferencia que existe entre los mismos.

Un **proyecto** es una colección de configuraciones para una instancia de **Django**, incluyendo configuración de base de datos, opciones específicas de **Django**, y configuraciones específicas de aplicaciones. Una **aplicación** es un conjunto portable de una funcionalidad de Django, típicamente incluye modelos y vistas que conviven en un solo paquete de Python, y que forman parte del proyecto. Una aplicación o módulo realiza una tarea específica, o un conjunto de ellas.

Django divide los proyectos en aplicaciones por las siguientes razones:

- Apoya la división del trabajo en equipo.
- Apoya el DDD (Domain Driven Design, Diseño Guiado por el Dominio).
- Apoya el concepto de Convención y Configuración.
- Dada la normalización, facilita el mantenimiento.
- Las aplicaciones pueden verse como Plugins.

- Apoya el concepto de SRP (Single Responsability, Principio de responsabilidad única).
- Estructurar la aplicación no es una preocupación.
- Mejora la reducción de errores por la descomposición en pequeños componentes.
- Se aplica para trabajar proyectos grandes y complejos.

Django sigue un patrón de diseño MVC, e impone una estructura MTV determinada que hay que seguir:

1. Todas las vistas deben ser colocadas bajo la carpeta Views, y nombrado con el sufijo View.
2. Todos los controles deben ser colocados bajo la carpeta Controllers. y nombrado con el sufijo Controller.

En Django se puede implementar todo un proyecto con una sola aplicación, y un único controlador, pero el mantenimiento será más difícil y no conveniente.

Las aplicaciones permiten trabajar tus proyectos como compuestos por plugins, es decir, directorios autónomos. Puedes incluir, o no, un app con solo agregarlo o eliminarlo en el archivo de configuración (settings). Además, obliga a descomponer tu solución en módulos, lo cual es conveniente, en especial cuando se utiliza lenguajes interpretados, ya que puedes detectar y corregir errores fácilmente.

EXERCISE 3: EJEMPLO DE PROYECTO Y APLICACIONES

Podemos tener un proyecto para la gestión de **compras en línea**.

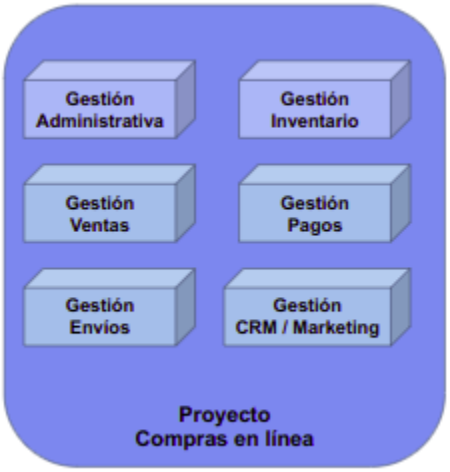

Figura 2.

Procedemos a crear el proyecto en Django. Por ejemplo, **compras_en_linea** como se ve en la Figura 2.

Supongamos que deseamos crear una aplicación para el proyecto, y la primera aplicación es la **Gestión Administrativa,** que maneja el inventario, los clientes, las ventas realizadas, el ingreso por compras, entre otras.

Seguidamente, podemos tener una segunda aplicación que **gestiona el inventario** (Stock de Inventario), con el fin de gestionar cuántos son los productos en el inventario real, las fallas con relación al inventario por falta de productos, gestión de productos con relación a un tiempo determinado (día, mes).

Así mismo, en el proyecto de compras en línea y de su complejidad, podemos tener un conjunto de aplicaciones. Por ejemplo: **gestión de ventas**, **gestión de pagos**, y medio de pago. Seguidamente, podemos también agregar una aplicación de **gestión de envíos**, y finalmente una que se encargue de **gestionar el marketing** o un pequeño CRM que gestione las relaciones con el cliente, basada en su satisfacción y promociones.

Generalmente, la creación de un proyecto con sus aplicaciones sigue el enfoque de la ***Programación Orientada a Objetos***.

Saber y conocer cuántas aplicaciones debo crear en un proyecto, depende de la complejidad, imaginación, habilidad, diseño que se tenga del proyecto en sí, y de la destreza en la práctica para implementarlas.

Es decir, para el proyecto de compras en líneas según las necesidades del cliente, un equipo de desarrollo la puede implementar con 8 aplicaciones, y otro equipo de desarrolladores la implementa con 6; finalmente satisface las necesidades del proyecto en sí. Esto se basa en el concepto de la modularización en Python.

La ventaja que tiene esta estructura de Django basada en proyecto y aplicación, es que se fundamenta en el principio de la reutilización.

EXERCISE 4: EJEMPLO DEL PRINCIPIO DE LA REUTILIZACIÓN

Suponemos que se desea implementar una aplicación Web que gestione el inventario de productos desde el navegador. Procedemos a crear un proyecto llamado **Gestión de Inventario,** con un determinado número de aplicaciones que se basan según su complejidad.

Para la gestión de inventario deseamos revisar los **proveedores** que surten de productos al inventario; en este sentido, gestionamos desde esta aplicación lo concerniente a proveedores.

Seguidamente, para este mismo proyecto, necesitamos una aplicación que gestione el stock de inventario con relación a los productos, disponibilidad, fallas, salidas, entradas de productos. En este caso podemos reutilizar la aplicación que se usó para el proyecto **compras en línea**, brindando la estructura de Django la reutilización de aplicaciones entre proyectos a pesar de que sean distintos. Este principio no es más que la modularización en Python, que se refiere a dividir un proyecto en distintas aplicaciones, lo que permite la escalabilidad superior llamada reutilización.

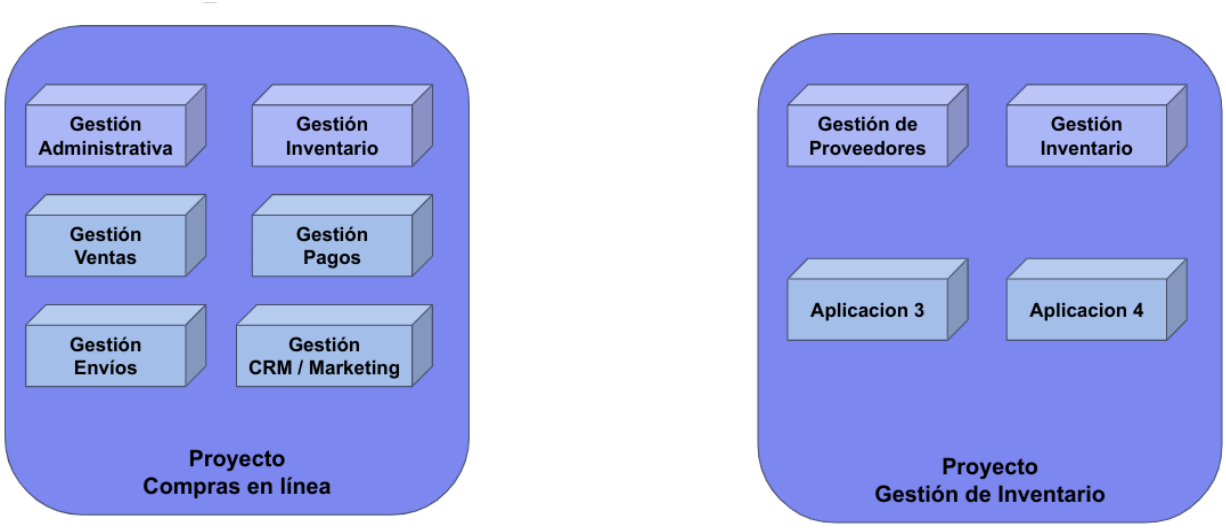

Figura 3.

EXERCISE 5: EJEMPLO DE APLICACIÓN EN DJANGO EN PROYECTO

Para observar más en detalle cómo se aplica esto de proyecto/aplicación en Django, lo detallamos de manera sencilla en el siguiente ejemplo basado en el sistema de **compras en línea**.

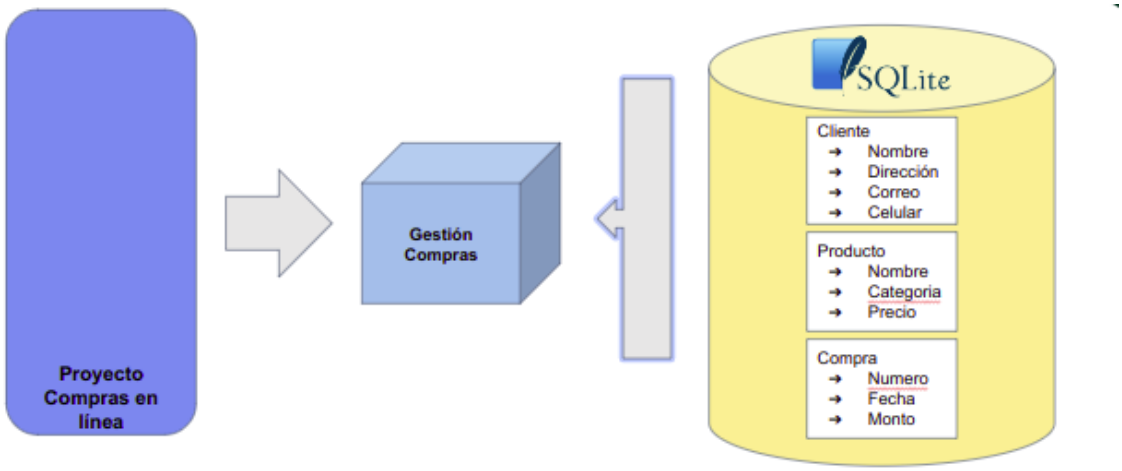

Figura 4.

El proyecto contiene una aplicación basada en la gestión de compras. En este sentido, el sistema de compras en línea gestiona las ventas a los clientes, tiene el inventario de productos, y una aplicación de gestión de compras por parte del cliente. El cliente entra al sistema de compras en línea, verifica si está disponible el producto para la compra en el inventario, y se emite una orden de compra por medio de un determinado número de compra.

Es decir, un determinado cliente realiza la compra, la aplicación genera la orden de compra con un número 55 con una fecha del 07 de agosto de 2022, y ésta se genera.

Para esto, la aplicación de **gestión de compras** por parte del cliente necesita un conjunto de modelos que se basan en una base de datos. Esto se realiza por un manejador de base de datos, por ejemplo SQLite3, y dentro de ella tendremos las siguientes tablas: Clientes, Productos y Compras, con los atributos que se reflejan en cada una de las tablas, tal como se muestra en la figura 4.

En este sentido, un cliente determinado selecciona los productos a comprar y se genera la orden de compra.

Finalmente, este es el esquema básico que integrará la aplicación **gestión de compras** en el proyecto de **compras en línea.**

EXERCISE 6: CREACIÓN DE LAS TABLAS EN UNA BASE DE DATOS

En Django, la creación de las tablas se basa a través de la Clase Model, que encuentra estructura en su lógica, las herramientas, funciones, propiedades, entre otros, y permite manejar la estructura de una base de datos en Django. Por ejemplo: la creación de tablas, eliminar tablas, crear campos en las tablas, eliminarlos y modificarlos, y especificar el tipo de datos de los atributos de las tablas que se encuentran dentro de la Clase Model.

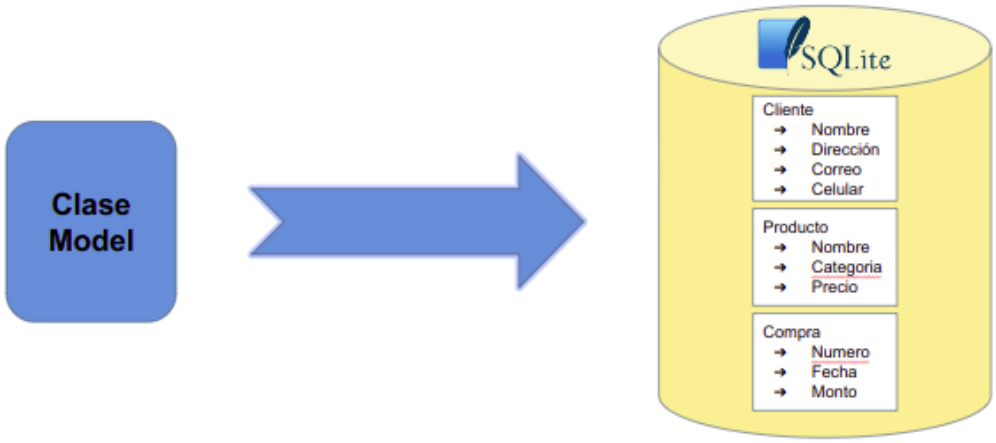

Figura 5.

Dentro de Django es importante tener en cuenta la estructura entre proyectos y aplicaciones, ya que éste no puede trabajar con modelos si no se ha creado previamente una aplicación para el proyecto, pues están íntimamente vinculados.

Por ejemplo, para crear las tablas de Cliente, Producto y Compra, se debe construir o crear una aplicación, no se puede crear una estructura independiente en base de datos, ya que no se va a relacionar con la gestión que realiza la Clase Model. Es decir, la Clase Model necesita de una aplicación para poder gestionar las bases de datos.