

ÍNDICE

ESPECIFICACIONES.....	4
INTRODUCCIÓN.....	4
INTERFACES.....	5
FUNCIONALIDADES.....	5
OBJETO SELF.....	6
OTRAS FUNCIONES.....	8
BASE DE DATOS.....	9
REALIZACIÓN.....	12
CONCLUSIÓN.....	18

ESPECIFICACIONES

Se deberá representar una empresa que está desarrollando un módulo para el ERP de un cliente. El cliente es un supermercado que tiene una base de datos de control de clientes, productos y pedidos con las tablas: categoría, cliente, producto, pedido y detalle.

- El módulo debe contar con una interfaz de usuario realizada con Tkinter en la que se dispongan los elementos necesarios para realizar todas las acciones especificadas por el cliente. Podrá utilizarse una única pantalla o varias, siempre que queden reflejadas todas las especificaciones necesarias.
Opcionalmente se utilizarán menús y se dispondrán los elementos de la UI con un diseño de aplicación profesional y/o se utilizarán elementos no vistos en clase.
- El módulo deberá conectarse a la base de datos del cliente con Sqlite3 y permitirá añadir nuevos ítems a las tablas y actualizar los existentes.
Opcionalmente, permitirá borrar registros y tablas y crear nuevas tablas.
- Deberá permitir realizar consultas y ordenar todas las tablas por cualquier campo (p. ej. el stock de un producto, los clientes de una ciudad, la fecha de un pedido, precios,...) y mostrar los resultados en forma de lista.
Opcionalmente se puede buscar la manera de realizar filtros condicionales (por ejemplo para buscar productos de los que se tenga un stock inferior a 3 ud. o con precio superior a x €)
Opcionalmente permitirá exportar la consulta a un archivo excel
- Deberá permitir mostrar también las consultas en forma de gráfico (se puede elegir el formato de gráfico que se crea más conveniente).
Opcionalmente se utilizará un formato de gráfico personalizado (y distinto al que se haya elegido como genérico) para al menos dos consultas.
Opcionalmente se permitirá al usuario elegir entre tres formatos de gráfico.

INTRODUCCIÓN

En este proyecto, asumimos el rol de desarrollador para una empresa que está creando un módulo para el ERP de un cliente, un supermercado. El cliente cuenta con una base de datos que gestiona información vital para su operación.

El objetivo del módulo es proporcionar una interfaz de usuario intuitiva utilizando Tkinter. Esta interfaz deberá permitir al usuario realizar diversas acciones, desde añadir y actualizar registros hasta la posibilidad opcional de eliminar registros y tablas, así como la creación de nuevas tablas.

La conectividad con la base de datos del cliente se realizará mediante Sqlite3, y el módulo permitirá realizar consultas y ordenar todas las tablas por cualquier campo, ofreciendo flexibilidad en la presentación de la información.

En resumen, el proyecto busca desarrollar un módulo eficiente, versátil y estéticamente atractivo que cumpla con las necesidades específicas del cliente y ofrezca características opcionales avanzadas para mejorar la experiencia del usuario.

Pero eso sí, no nos ha salido todo perfecto. Quiero decir SI, ha salido bastante bien.

Me guarda todo en la tabla lo cual es increíble y perfecto. Hay que tener cuidado con los Entry que pueden dar muchos errores.

INTERFACES

Nuestro trabajo consta de tres interfaces.

1º INTERFAZ: INICIO

La primera interfaz, será la página principal, donde se dará la bienvenida al usuario. En ella encontraremos el logo de la empresa y su nombre. Además de información sobre la misma.

Esta página tendrá un Menú arriba de ella en la que tendremos 2 posibles opciones: COMPRAS y SALIR. Si marcamos COMPRAS nos saldrán dos opciones más, PRODUCTOS y CARRITO.

2º INTERFAZ: PRODUCTOS

La segunda interfaz PRODUCTOS, nos aparecerá la fecha actual y los productos que podremos comprar. AL pinchar en ellas, debemos ir a la interfaz CARRITO

3º INTERFAZ: CARRITO

Cuando hayamos elegido nuestros productos y nos vamos a la interfaz CARRITO. Nos aparecerá el número de productos elegidos, además del nombre de ellos.

Posteriormente tendremos que añadir, en el apartado cliente nuestro nombre y apellido, teléfono, código postal y nuestra forma de pago.

Luego debemos añadir el apartado de producto, con su categoría, tamaño y precio.

Cuando hayamos terminado, le daremos a COMPRAR, para terminar y nos debería de salir un mensaje diciendo que se ha guardado correctamente.

FUNCIONALIDADES

FUNCIONALIDADES, no es el número de interfaces. Aquí daremos más detalles sobre cómo funciona la aplicación.

Para empezar, en la Interfaz INICIO tendremos dos submenús, como ya hemos dicho.

SALIR y COMPRAS. Al darle a SALIR cerrará la aplicación. Pero si dejamos marcado COMPRAS nos aparecerá PRODUCTOS y CARRITO:

Al darle a PRODUCTOS nos parecerán los títulos de los productos, con imágenes que se pueden clicar, si nos gusta alguna (elegir uno para probar funcionalidad) y que debes hacerlo. Luego en PRODUCTOS también hay un submenú, INICIO y COMPRAS > CARRITO.

Si le damos a INICIO, nos borrará la página PRODUCTOS. Por otra parte, si seleccionamos COMPRAS y le damos a CARRITO. Nos abrirá CARRITO.

Ya en la interfaz CARRITO, nos aparecerá el número de clics que hemos dado a las imágenes de la interfaz PRODUCTOS y sus respectivos nombres, como si fueran los nombres de los productos.

Abajo tendrás que rellenar los apartados ya mencionados en NÚMERO DE INTERFACES.

Y por último y más importante, estará el botón COMPRAR. Si le damos al botón COMPRAR, se nos guardarán los datos añadidos a nuestra base de datos y nos aparecerá un mensaje diciendo que la compra se ha realizado correctamente o un mensaje de error, si no se han podido guardar. Si quieres añadir otros productos, debes ir en orden. Primero añades el primer producto, le das a COMPRAR y luego añades otro, así.

OBJETO SELF

self es una práctica común que se utiliza para referirse al objeto actual de una clase. En los métodos de una clase, el primer parámetro siempre se llama self, aunque se pueda llamar de otra manera, como ya he dicho es una práctica común.

Cuando llamas a un método en un objeto de una clase, Python automáticamente pasa la instancia del objeto como el primer argumento al método. De esta manera, puedes acceder a los atributos y métodos de esa instancia dentro del método utilizando la palabra clave self.

```
class MiClase:

    def __init__(self, atributo):

        self.atributo = atributo


    def metodo(self):

        print(f"Este es un método de la instancia. Atributo:
{self.atributo}")


# Crear una instancia de la clase
objeto = MiClase("Hola")


# Llamar al método en la instancia
```

`objeto.metodo()`

En este ejemplo:

1. `objeto` es una instancia de la clase `MiClase`.
2. Cuando llamamos al método `metodo` en `objeto`, no necesitamos pasar directamente la instancia como un argumento. Python lo hace automáticamente y lo asigna al parámetro `self` dentro del método.
3. Dentro de `metodo`, podemos acceder al atributo `atributo` de la instancia utilizando `self.atributo`.

Usar `self` es una forma de mantener el estado de las instancias y permite que los métodos de la clase operen en ese estado. Además, ayuda a distinguir entre variables locales y variables de instancia dentro de los métodos de la clase.

Esto lo menciono aquí ya que en mi código estaré utilizando `self` muy a menudo.

Cada vez que creamos una clase, e iniciemos cualquier variable en el método `__init__(self)`

Esa variable se va a poder utilizar en cualquier parte de la clase, siempre y cuando tenga el `self`:

```
class Carrito:

    def __init__(self):

        self.contador = 0
        self.productos_seleccionados = []
        self.cliente_nombre = ""
        self.cliente_apellidos = ""
        self.cliente_telefono = ""
        self.cliente_codigop = ""
        self.producto_categoria = ""
        self.producto_tamaño = ""
        self.producto_precio = ""
        self.producto_formaPago = ""
        self.numero = 0
        self.letras = 'ABCDEFGHIGKLMNÑOPQRSTUVWXYZ'
        self.nombreProducto = ""
        self.variable = [StringVar(), StringVar(), StringVar(),
                        StringVar(), StringVar(), StringVar(),
                        StringVar(), StringVar()]
```

Como se puede ver aquí.

Aquí por ejemplo, para mostrar los productos:

```
for self.nombreProducto in self.productos_seleccionados:
```

```

producto_label = tk.Label(carrito, text=self.nombreProducto,
font=('Roboto Mono', 12))
producto_label.pack()

```

Añadimos las variables con self pero como ya están inicializadas, con la palabra self, puede no hace falta darle un resultado de primeras.

Ahora, ¿Porque he elegido hacer clases? Pues esto ha sido para diferenciar las funciones de la clase Productos, de la clase o interfaz Carrito. Así está todo mejor estructurado

OTRAS FUNCIONES

Por ejemplo el uso de Stringvar, para relacionarla con el Entry:

```

self.variable = [StringVar(), StringVar(), StringVar(),
StringVar(), StringVar(), StringVar(),
StringVar(), StringVar()]

```

Esta función lo que hace es vincularse con una entrada de texto, en este caso, aunque puede ser con texto normal. Y una vez vinculados, cuando Entry se actualice o cambie, automáticamente StringVar() se actualizará igualmente. Por eso la utilizamos, para guardarla en la base de datos las respuestas de los usuarios en los Entry.

¿Pero como sabias cual era cual? te estarás preguntando. Pues porque abajo, donde colocamos los Entry los colocamos:

```

self.cliente_nombre = tk.Entry(colocacion_variables,
textvariable=self.variable[0], width=30)

```

con esto == > **self.variable[0]** . Como es una lista pues lo almacena.

Tambien tenemos los Frame, que son contenedores, donde guardar todo tipo de datos:

```

colocacion_variables = tk.Frame(carrito)
colocacion_variables.pack(pady=10, padx=50)

```

Aqui se añade a carrito o productos, la que sea, o ventana. Y con pack lo añades. SIN es PACK no se añade a la ventana, no aparece directamente.

Luego añades cosas, y en vez de utilizar pack y ventana utilizas el nombre del Frame y grid.

```

self.cliente_nombre = tk.Entry(colocacion_variables,
textvariable=self.variable[0], width=30)

```

```
self.cliente_nombre.grid(row=1, column=1, padx=5, pady=5, sticky=tk.W)
```

Ahí se utiliza `colocacion_variables`, con `width` hacemos más grande la barra de entrada de texto. Y en el código de abajo `row` es como si fuera = fila y `column` = columna, aparte de `padx` = espacio en horizontal y `pady` = espacio en vertical. Además de `Sticky` que es para colocar el dato al lado oeste de la celda, a la izquierda.

También la utilización de `tk.Menu` para hacer los menús de arriba. Que con `add_cascade`

colocas el principal, el que primero se verá y `add_command` para poner lo secundarios, que cuando clickes en el principal te saldrán marcados. Además está `add_separator()` para separarlos por líneas, como veremos en el trabajo.

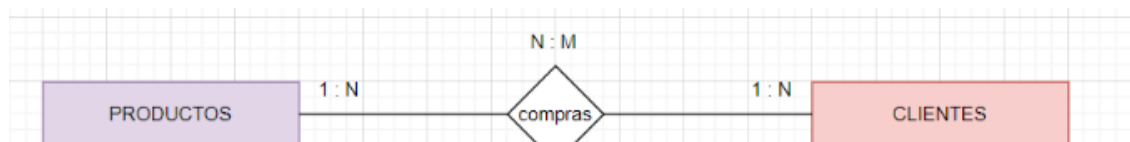
Con esto ya más o menos lo tienes todo.

BASE DE DATOS

1º CREACIÓN DE BBDD.

Lo primero que haremos, o de las primeras cosas que haremos será crear nuestra bbdd.

Con tres tablas para ser más específicos. Productos - Clientes - Compras, que Compras llevará la clave primaria de Productos y Clientes.



Al tener las dos tablas anteriores números infinitos asociados. (En Productos he puesto 1:N, ya que aunque ahora solo haya cuatro productos, aunque en realidad hay seis, seguirán viniendo más)

En la Tabla Productos, tendremos de PK (Primary Key) a `Cod_Producto` (Código Producto) que será un número de 1 a 4 y seguirá subiendo, además de otros atributos como el nombre, la categoría, el tamaño y el precio.

En la Tabla Clientes, tendremos como PK a `ID_cliente` que será con letras al azar, además de Nombre, Apellidos, Tlf(Telefono) y Código Postal.

Por último crearemos la Tabla Compras como PK las PKs de Productos y Clientes, es decir, `Cod_Producto` y `ID_cliente`. Además de ser también sus FK (Foreign Key). También podemos encontrar `Forma_pago`, ósea forma de pago y Fecha.

Como podemos ver en el código creamos la conexión con la

```
# Crear una conexión a la base de datos
conexion = sqlite3.connect("bbddParquejody.db")
```

```
cursor = conexion.cursor()
```

Como podemos ver en el código, creamos la conexión con la base de datos con `sqlite3`, que arriba deberemos importarla para que podamos conectarnos o crear una bbdd. Nos conectamos con `Connect`, que como también está ese método en Java, sirve para dar a conocer a la bbdd que nos estamos conectando a ella.

Después creamos un objeto cursor asociado con la conexión a la base de datos.

Un cursor es un objeto que permite interactuar con la base de datos, ejecutar consultas SQL y recuperar resultados. Este objeto nos será de gran ayuda en este proyecto.

Luego ya creamos las tablas como si estuviéramos en una aplicación SQL normal, pero con el cursor y el método `execute`. Este método se usa para ejecutar sentencias SQL de bases de datos, a través de un cursor. Exactamente así.

```
# Crear la tabla Clientes
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Clientes (
        Id_cliente TEXT PRIMARY KEY,
        Nombre TEXT,
        Apellido TEXT,
        Tlf TEXT,
        CodigoP INTEGER
    )
''')

# Crear la tabla Compras
cursor.execute('''
    CREATE TABLE IF NOT EXISTS Compras (
        Cod_Producto INTEGER NOT NULL,
        Id_cliente TEXT NOT NULL,
        Forma_pago TEXT,
        Fecha TEXT NOT NULL,
        FOREIGN KEY(Id_cliente) REFERENCES Clientes(Id_cliente),
        FOREIGN KEY(Cod_Producto) REFERENCES Productos(Cod_Producto),
    )
''')
```



```

        PRIMARY KEY(Id_cliente, Cod_Producto)

    )

'''

# Crear la tabla Productos

cursor.execute('''

    CREATE TABLE IF NOT EXISTS Productos (

        Cod_Producto INTEGER PRIMARY KEY,

        Nombre TEXT,

        Categoria TEXT,

        Tamaño TEXT,

        Precio NUMERIC

    )

''')

```

Luego a la hora de añadir los datos a la base de datos será de la siguiente manera:

```

cursor.execute("INSERT INTO Clientes VALUES (?, ?, ?, ?, ?)",
(self.letrasAleatorios(), m, self.cliente_apellidos,
self.cliente_telefono, self.cliente_codigop))

cursor.execute("INSERT INTO Productos VALUES (?, ?, ?, ?, ?)",
(self.numero, self.nombreProducto ,self.producto_categoria,
self.producto_tamaño, self.producto_precio))

cursor.execute("INSERT INTO Compras VALUES (?, ?, ?, ?)",
(self.numero, self.letrasAleatorios(), self.producto_formaPago,
datetime.now().strftime("%d-%m-%Y")))

```

Aquí básicamente lo que hago es añadir los datos obtenidos a lo largo de todo el código en nuestra base de datos creada anteriormente.

Dónde están las interrogaciones, añadiremos una por una los datos. Es MUY IMPORTANTE, añadirlas en orden de cómo lo has creado, si no se te puede descuadrar todo y dar error, al no cuadrarse con los tipos de datos de la tabla.

```
conexion.commit() # Guardar los cambios en la base de datos
```

Este metodo para actualizar la base de datos una vez añadamos los datos.

```
cursor.execute("SELECT * FROM Productos")

result_compra = cursor.fetchall()

print("Datos insertados en la tabla Productos:", result_compra)
```

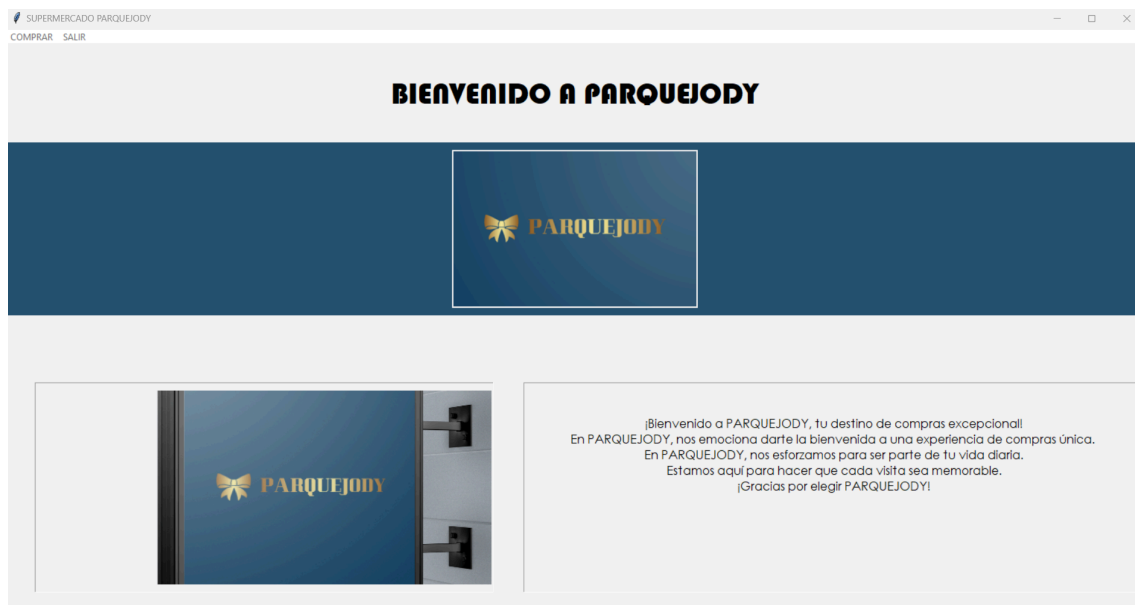
Por último, en relación con la bbdd, vamos a comprobar si se han guardado los cambios en la tabla. Con el `cursor` y el `execute` inicializando una consulta, para mostrar todos los datos de la Tabla `Productos`.

Luego creamos una variable y le decimos que el cursor, el ejecutor de consultas ejecuta línea por línea, con `fetchall()`. Y así lo hace, mostrándonos por consola el código:

```
PS C:\Users\marqu\OneDrive\Documentos\PYTHON\HITO2> & C:/Users/marqu/anaconda3/python.exe c:/Users/marqu/OneDrive/Documentos/PYTHON/HITO2/CrearPagina/APLICACIÓN
Datos insertados en la tabla Productos: [(1, 'MacBook Pro Touch Bar 15 - 855.44€', 'Tecnologica - Casa', 'Pequeño - Mediano - Grande', '€')]
PS C:\Users\marqu\OneDrive\Documentos\PYTHON\HITO2> █
```

REALIZACIÓN

Al principio, al abrir la pagina nos saltará la primera Interfaz INICIO:



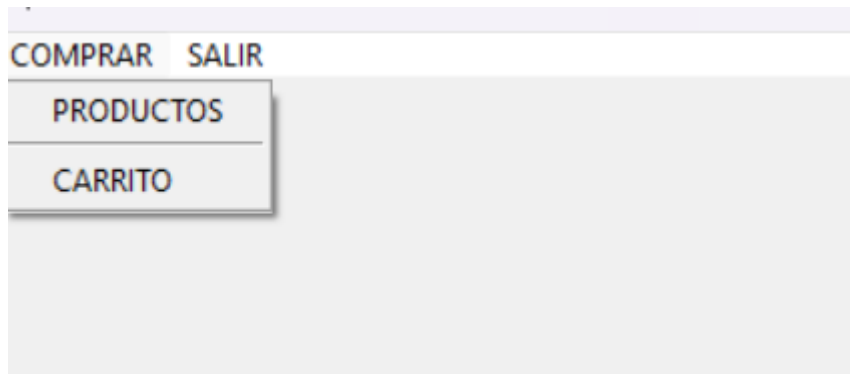
No es la más bonita de las páginas, pero como no domino Tkinter pues he creado esto.

Luego, como hemos dicho, tenemos un Menú arriba, donde podemos seleccionar `SALIR` o `COMPRAS`.

La opción salir de la aplicación:



O La opción para irnos ha las otras interfaces:



Si en principio nos vamos a CARRITO, no sabremos qué productos añadiremos:

Numero de productos: 0

Productos en el carrito:

ID Cliente: GINFN

Nombre:

Apellidos:

Telefono:

Codigo P.:

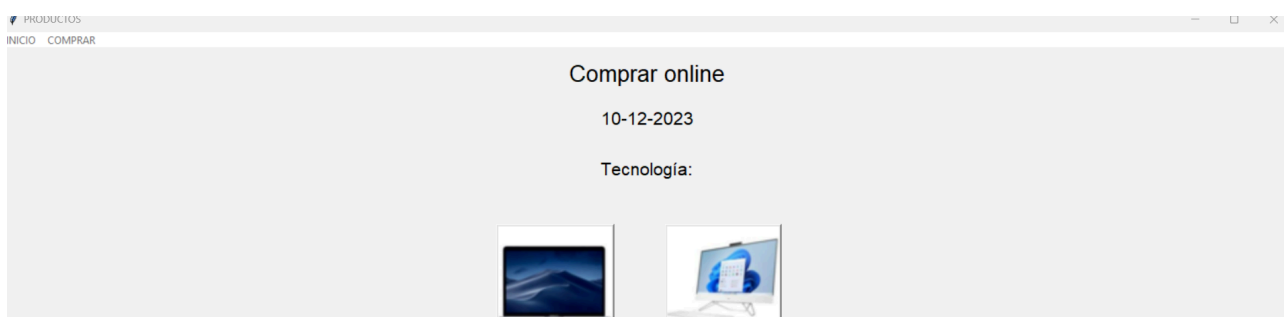
CATEGORIA:

TAMAÑO:

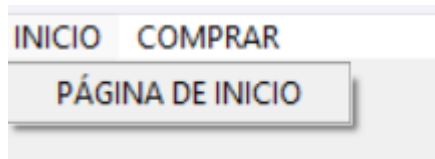
PRECIO:

FORMA DE PAGO:

Pero si lo hacemos bien y primero le das a PRODUCTOS, se abrirá la página:



En la cual podremos ir al INICIO, la primera Interfaz.



Aún hay pocos productos, ya que el supermercado está empezando.

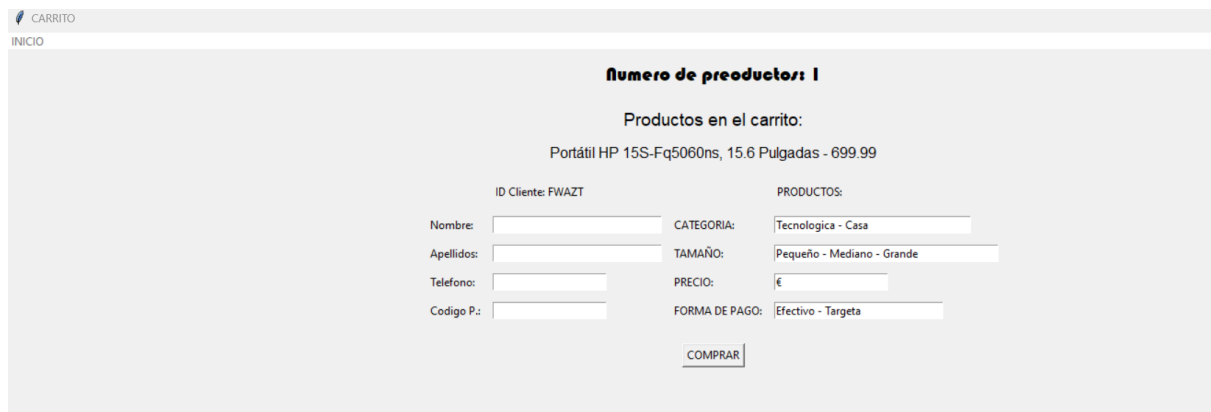
Si clicas uno o varios de ellos (los productos se ven borrosos por hacer las imágenes más pequeñas en otra web)



Y ahora si vamos a CARRITO



Se nos abra la última interfaz, con sus datos añadidos. El nombre del producto con su precio, para que lo añadas, y el número de productos que has añadido al carrito:



Puedes añadir todos los productos que quieras, que se van a colocar y se va a ir aumentando el número de productos.

Numero de productos: 10

Productos en el carrito:
MacBook Pro Touch Bar 15 - 855.44€
MacBook Pro Touch Bar 15 - 855.44€
MacBook Pro Touch Bar 15 - 855.44€
Portátil HP 15S-Fq5060ns, 15.6 Pulgadas - 699.99
Portátil HP 15S-Fq5060ns, 15.6 Pulgadas - 699.99
Portátil HP 15S-Fq5060ns, 15.6 Pulgadas - 699.99
Conjunto De Muebles De Ratán 135 X 72.5 X 62cm Color Gris - 536€
Conjunto De Muebles De Ratán 135 X 72.5 X 62cm Color Gris - 536€
Conjunto De Muebles De Ratán 135 X 72.5 X 62cm Color Gris - 536€
Mueble De Entrada De 2 Puertas Con Espejo - 207€

ID Cliente: DHBÑS

PRODUCTOS:

Nombre:

CATEGORIA:

Apellidos:

TAMAÑO:

Telefono:

PRECIO:

Codigo P.:

FORMA DE PAGO:

Si
te

fijas, en la parte productos, te especifica un poco lo que tienes que poner para no perderte. Además como hay tan pocos productos de momento pues es beneficioso para el cliente añadir los datos.

Además, cada vez que vuelvas a CARRITO tu ID como cliente irá cambiando.

Ahora, es hora de añadir los datos:

Numero de productos: 1

Productos en el carrito:
Portátil HP 15S-Fq5060ns, 15.6 Pulgadas - 699.99

ID Cliente: FWAZT

PRODUCTOS:

Nombre:

CATEGORIA:

Apellidos:

TAMAÑO:

Telefono:

PRECIO:

Codigo P.:

FORMA DE PAGO:

Si le damos al botón guardar y hay algún tipo de error nos saltará este mensaje:

Numero de productos: 1

Productos en el carrito:
Portátil HP 15S-Fq5060ns, 15.6 Pulgadas - 699.99

ID Cliente: FWAZT

Nombre:	Marcos	CATEGORIA:	Tecnologica
Apellidos:	Jodar Gomez	TAMAÑO:	Pequeño
Telefono:	646253088	PRECIO:	699.99 €
Codigo P.:	28914	FORMA DE PAGO:	Efectivo

Error


Error al realizar la compra: datatype mismatch

Aceptar

Pero si esta todo en orden, y nos añade los datos a la tabla

Nos aparecerá el mensaje que esperamos.

BIENVENIDO A PARQUEJODY

**PARQUEJODY**

COMPRA

Tu compra fue realizada

Aceptar

¡Bienvenido a PARQUEJODY, tu d
En PARQUEJODY, nos emociona darte la bien
En PARQUEJODY, nos esforzamos
Estamos aquí para hacer que
¡Gracias por elegi