



**UAEM**

Base de datos

Centro Universitario UAEM Atlacomulco.

Universidad Autónoma del Estado del México.

Licenciatura en ingeniería en Computación.

Tercer Semestre

Otoño 2024.

Maestro. Julio Alberto De la teja Lopez

Alumno: Alan Osornio Garcia

Jorge Luis gil Bello

Emanuel Nieto Garcia.

ICO 28.

25/11/2024.

## Descripción General

Este es un programa web construido con **Flask** que permite gestionar un sistema de productos. Las operaciones admitidas incluyen:

- Listar productos existentes.
- Agregar nuevos productos.
- Editar información de productos.
- Eliminar productos.

El programa se conecta a una base de datos MySQL llamada `sales_cart` y utiliza una tabla llamada `products` para almacenar información sobre los productos.

---

## Estructura del Proyecto

El proyecto sigue una estructura básica de Flask con las siguientes características principales:

### 1. Rutas (Endpoints):

- `/`: Muestra todos los productos.
- `/add`: Permite agregar un nuevo producto.
- `/edit/<int:id>`: Permite editar un producto existente.
- `/delete/<int:id>`: Elimina un producto basado en su ID.

### 2. Templates (Archivos HTML):

- `products.html`: Para listar todos los productos.
  - `add_product.html`: Para agregar un nuevo producto.
  - `edit_product.html`: Para editar un producto.
- 

## Requisitos Previos

### 1. Instalar las Dependencias:

Asegúrate de tener instalado Flask y el conector de MySQL. Usa los siguientes comandos:

```
bash
Copiar código
pip install flask mysql-connector-python
```

### 2. Configurar la Base de Datos:

Crea la base de datos y la tabla `products` en MySQL:

```
sql
```

```
Copiar código
CREATE DATABASE sales_cart;

USE sales_cart;

CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);
```

---

## Código Detallado

### 1. Conexión a la Base de Datos

La función `get_db_connection` se encarga de establecer la conexión con la base de datos MySQL.

```
python
Copiar código
def get_db_connection():
    return mysql.connector.connect(
        host='localhost',
        user='root',
        password='', # Si no tienes contraseña, déjalo vacío
        database='sales_cart'
)
```

---

### 2. Rutas

#### a) Mostrar Productos

Ruta principal (/) que lista todos los productos en la tabla `products`.

- Método HTTP: GET.
- Lógica:
  - Realiza una consulta SQL para obtener todos los registros.
  - Renderiza `products.html` y pasa los datos de los productos.

```
python
Copiar código
@app.route('/')
def show_products():
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM products;")
    products = cursor.fetchall()
    cursor.close()
    connection.close()
    return render_template('products.html', products=products)
```

---

### b) Agregar Producto

Ruta para agregar un nuevo producto (/add).

- Métodos HTTP: GET (formulario) y POST (guardar).
- Lógica:
  - Renderiza un formulario HTML con add\_product.html para capturar datos (GET).
  - Inserta un nuevo producto en la tabla products al enviar el formulario (POST).

```
python
Copiar código
@app.route('/add', methods=['GET', 'POST'])
def add_product():
    if request.method == 'POST':
        name = request.form['name']
        price = request.form['price']

        connection = get_db_connection()
        cursor = connection.cursor()
        cursor.execute("INSERT INTO products (name, price) VALUES (%s,
%s)", (name, price))
        connection.commit()
        cursor.close()
        connection.close()

    return redirect(url_for('show_products'))

    return render_template('add_product.html')
```

---

### c) Editar Producto

Ruta para editar un producto existente (/edit/<int:id>).

- Métodos HTTP: GET (formulario con datos actuales) y POST (guardar cambios).
- Lógica:
  - Consulta un producto por su ID y muestra el formulario prellenado (GET).
  - Actualiza el producto en la base de datos después de editarlo (POST).

```
python
Copiar código
@app.route('/edit/<int:id>', methods=['GET', 'POST'])
def edit_product(id):
    connection = get_db_connection()
    cursor = connection.cursor(dictionary=True)
    cursor.execute("SELECT * FROM products WHERE id = %s", (id,))
    product = cursor.fetchone()
```

```
if request.method == 'POST':
    name = request.form['name']
    price = request.form['price']

    cursor.execute("UPDATE products SET name = %s, price = %s WHERE
id = %s", (name, price, id))
    connection.commit()
    cursor.close()
    connection.close()

    return redirect(url_for('show_products'))

cursor.close()
connection.close()
return render_template('edit_product.html', product=product)
```

---

#### d) Eliminar Producto

Ruta para eliminar un producto basado en su ID (/delete/<int:id>).

- Método HTTP: POST.
- Lógica:
  - Elimina un producto usando su ID.
  - Redirige de vuelta a la lista de productos.

```
python
Copiar código
@app.route('/delete/<int:id>', methods=['POST'])
def delete_product(id):
    connection = get_db_connection()
    cursor = connection.cursor()
    cursor.execute("DELETE FROM products WHERE id = %s", (id,))
    connection.commit()
    cursor.close()
    connection.close()

    return redirect(url_for('show_products'))
```

---

### Templates HTML (Ejemplo)

**products.html**

Un ejemplo básico para mostrar los productos:

```
html
Copiar código
<!DOCTYPE html>
<html>
<head>
    <title>Lista de Productos</title>
</head>
```

```

<body>
    <h1>Lista de Productos</h1>
    <a href="{{ url_for('add_product') }}>Agregar Producto</a>
    <table border="1">
        <tr>
            <th>ID</th>
            <th>Nombre</th>
            <th>Precio</th>
            <th>Acciones</th>
        </tr>
        {% for product in products %}
        <tr>
            <td>{{ product.id }}</td>
            <td>{{ product.name }}</td>
            <td>{{ product.price }}</td>
            <td>
                <a href="{{ url_for('edit_product', id=product.id) }}>Editar</a>
                <form action="{{ url_for('delete_product', id=product.id) }}" method="post" style="display:inline;">
                    <button type="submit">Eliminar</button>
                </form>
            </td>
        </tr>
        {% endfor %}
    </table>
</body>
</html>

```

---

## Cómo Ejecutar

1. Guarda el archivo como `app.py`.
2. Ejecuta el servidor Flask:

```

bash
Copiar código
python app.py

```

3. Abre tu navegador y navega a `http://127.0.0.1:5000/`.

Capturar del código

## Productos

Agregar nuevo producto

ID	Nombre	Precio	Acciones
2	Elden Ring	49.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
3	Cyberpunk 2077	39.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
4	Super Mario Odyssey	59.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
5	Minecraft	26.95	<a href="#">Editar</a> <a href="#">Eliminar</a>
6	Fortnite: Save the World	29.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
7	Call of Duty: Modern Warfare II	69.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
8	Red Dead Redemption 2	59.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
9	Grand Theft Auto V	29.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
10	Hogwarts Legacy	59.99	<a href="#">Editar</a> <a href="#">Eliminar</a>
11	FIFA 24	69.99	<a href="#">Editar</a> <a href="#">Eliminar</a>

Activar

## Editar producto

Nombre del producto:

Elden Ring

Precio:

49.99

[Guardar cambios](#)

[Volver a la lista de productos](#)

# Agregar nuevo producto

Nombre del producto:

Precio:

**Agregar producto**

[Volver a la lista de productos](#)

## Base de datos en xampp

The screenshot shows the MySQL Workbench interface. On the left, there's a tree view of databases and tables. The 'products' table under the 'sales\_cart' database is selected. The main pane displays the following SQL query at the top:

```
SELECT * FROM `products`
```

Below the query, there are several buttons: 'Perfilando' (disabled), 'Editar en línea', 'Editar', 'Explicar SQL', 'Crear código PHP', and 'Actualizar'. There are also filters for 'Mostrar todo', 'Número de filas: 25', 'Filtrar filas', 'Buscar en esta tabla', and 'Ordenar según la clave: Ninguna'. A 'Opciones extra' button is also present.

The 'products' table has the following columns: id, name, price, and stock. The data is as follows:

	id	name	price	stock
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	2	Elden Ring	49.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	3	Cyberpunk 2077	39.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	4	Super Mario Odyssey	59.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	5	Minecraft	26.95	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	6	Fortnite: Save the World	29.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	7	Call of Duty: Modern Warfare II	69.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	8	Red Dead Redemption 2	59.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	9	Grand Theft Auto V	29.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	10	Hogwarts Legacy	59.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	11	FIFA 24	69.99	0
<input type="checkbox"/> <a href="#">Editar</a> <a href="#">Copiar</a> <a href="#">Borrar</a>	12	Street Fighter 6	49.99	0

## 1. Rutas (Endpoints):

- `/`: Muestra los productos disponibles.
- `/add_to_cart/<int:product_id>`: Agrega un producto al carrito.
- `/cart`: Muestra los productos en el carrito.
- `/remove_from_cart/<int:item_id>`: Elimina un producto específico del carrito.
- `/checkout`: Finaliza la compra, crea una orden y vacía el carrito.

- /order\_confirmation/<int:order\_id>: Muestra un resumen de la orden.
2. **Templates HTML:**
- index.html: Lista los productos.
  - cart.html: Muestra el carrito con los productos seleccionados.
  - checkout.html: Confirma los detalles de la orden.
- 

## Requisitos Previos

### 1. Dependencias

Instala las librerías necesarias con:

```
bash
Copiar código
pip install flask mysql-connector-python
```

### 2. Configuración de la Base de Datos

Asegúrate de crear las tablas necesarias en tu base de datos MySQL. A continuación, un ejemplo del esquema:

```
sql
Copiar código
CREATE DATABASE sales_cart;

USE sales_cart;

-- Tabla de productos
CREATE TABLE products (
    id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    price DECIMAL(10, 2) NOT NULL
);

-- Tabla del carrito
CREATE TABLE cart (
    id INT AUTO_INCREMENT PRIMARY KEY,
    product_id INT NOT NULL,
    quantity INT NOT NULL DEFAULT 1,
    FOREIGN KEY (product_id) REFERENCES products(id)
);

-- Tabla de órdenes
CREATE TABLE orders (
    id INT AUTO_INCREMENT PRIMARY KEY,
    total_amount DECIMAL(10, 2) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabla de detalles de la orden
```

```
CREATE TABLE order_details (
    id INT AUTO_INCREMENT PRIMARY KEY,
    order_id INT NOT NULL,
    product_id INT NOT NULL,
    quantity INT NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

### 3. Configuración del Archivo config.py

Crea un archivo config.py con las credenciales de la base de datos:

```
python
Copiar código
import os

class Config:
    SECRET_KEY = os.urandom(24)
    MYSQL_HOST = 'localhost'
    MYSQL_USER = 'root'
    MYSQL_PASSWORD = '' # Cambiar según tu configuración
    MYSQL_DB = 'sales_cart'
```

---

## Detalles del Código

### 1. Conexión a la Base de Datos

La función get\_db\_connection conecta la aplicación a la base de datos MySQL usando las credenciales definidas en config.py.

```
python
Copiar código
def get_db_connection():
    return mysql.connector.connect(
        host=app.config['MYSQL_HOST'],
        user=app.config['MYSQL_USER'],
        password=app.config['MYSQL_PASSWORD'],
        database=app.config['MYSQL_DB']
)
```

---

## 2. Rutas

### a) Mostrar Productos (/)

Obtiene todos los productos de la base de datos y los muestra en una página.

```
python
Copiar código
```

```
@app.route('/')
def index():
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute('SELECT * FROM products')
    products = cursor.fetchall()

    cursor.close()
    connection.close()

    return render_template('index.html', products=products)
```

---

**b) Agregar al Carrito (/add\_to\_cart/<int:product\_id>)**

Agrega un producto al carrito o incrementa la cantidad si ya existe en el carrito.

```
python
Copiar código
@app.route('/add_to_cart/<int:product_id>', methods=['POST'])
def add_to_cart(product_id):
    quantity = request.form.get('quantity', 1, type=int)

    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("SELECT id, quantity FROM cart WHERE product_id = %s",
    (product_id,))
    cart_item = cursor.fetchone()
    cursor.fetchall() # Limpia resultados pendientes

    if cart_item:
        new_quantity = cart_item[1] + quantity
        cursor.execute("UPDATE cart SET quantity = %s WHERE id = %s",
        (new_quantity, cart_item[0]))
    else:
        cursor.execute("INSERT INTO cart (product_id, quantity) VALUES
        (%s, %s)", (product_id, quantity))

    connection.commit()
    cursor.close()
    connection.close()

    return redirect(url_for('cart'))
```

---

**c) Ver Carrito (/cart)**

Lista los productos agregados al carrito con sus cantidades y calcula el total.

```
python
Copiar código
@app.route('/cart')
```

```
def cart():
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("""
        SELECT p.name, p.price, c.quantity, c.id
        FROM cart c
        JOIN products p ON c.product_id = p.id
    """)
    cart_items = cursor.fetchall()

    total = sum(item[1] * item[2] for item in cart_items)

    cursor.close()
    connection.close()

    return render_template('cart.html', cart_items=cart_items,
total=total)
```

---

**d) Eliminar del Carrito (/remove\_from\_cart/<int:item\_id>)**

Elimina un producto específico del carrito.

```
python
Copiar código
@app.route('/remove_from_cart/<int:item_id>', methods=['POST'])
def remove_from_cart(item_id):
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("DELETE FROM cart WHERE id = %s", (item_id,))
    connection.commit()

    cursor.close()
    connection.close()

    return redirect(url_for('cart'))
```

---

**e) Checkout (/checkout)**

Finaliza la compra, crea una orden y vacía el carrito.

```
python
Copiar código
@app.route('/checkout', methods=['POST'])
def checkout():
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("""
        SELECT c.product_id, p.price, c.quantity
        FROM cart c
    """)
```

```

        JOIN products p ON c.product_id = p.id
      """
    cart_items = cursor.fetchall()

    total_amount = sum(item[1] * item[2] for item in cart_items)

    cursor.execute("INSERT INTO orders (total_amount) VALUES (%s)",
    (total_amount,))
    order_id = cursor.lastrowid

    for item in cart_items:
        cursor.execute("""
            INSERT INTO order_details (order_id, product_id, quantity,
            price)
            VALUES (%s, %s, %s, %s)
        """, (order_id, item[0], item[2], item[1]))

    cursor.execute("DELETE FROM cart")

    connection.commit()
    cursor.close()
    connection.close()

    return redirect(url_for('order_confirmation', order_id=order_id))

```

---

#### f) Confirmación de Orden (/order\_confirmation/<int:order\_id>)

Muestra los detalles de la orden recién creada.

```

python
Copiar código
@app.route('/order_confirmation/<int:order_id>')
def order_confirmation(order_id):
    connection = get_db_connection()
    cursor = connection.cursor()

    cursor.execute("SELECT id, total_amount FROM orders WHERE id = %s",
    (order_id,))
    order = cursor.fetchone()

    cursor.close()
    connection.close()

    return render_template('checkout.html', order=order)

```

---

## Cómo Ejecutar

1. Guarda el archivo como `app.py`.
2. Asegúrate de que `config.py` esté configurado correctamente.
3. Ejecuta el servidor Flask:

bash

Copiar código  
python app.py

4. Abre tu navegador y ve a <http://127.0.0.1:5000/>.

---

Este proyecto es un ejemplo funcional de un sistema básico de carrito de compras. Puedes personalizarlo y expandirlo según tus necesidades. ☺

Capturas de la pagina

The screenshot shows a dark-themed e-commerce website for video games. At the top, a blue header bar displays the title "Pec Shoc - Tienda de Videojuegos". Below the header, a section titled "Productos" lists ten video games in a 2x5 grid. Each game card includes the game's name, price, a quantity selector (set to 1), and a green "Añadir al carrito" (Add to cart) button.

Imagen	Título	Precio	Quant.	Botón
Elden Ring	Elden Ring	\$49.99	1	Añadir al carrito
Cyberpunk 2077	Cyberpunk 2077	\$39.99	1	Añadir al carrito
Super Mario Odyssey	Super Mario Odyssey	\$59.99	1	Añadir al carrito
Minecraft	Minecraft	\$26.95	1	Añadir al carrito
Fortnite: Save the World	Fortnite: Save the World	\$29.99	1	Añadir al carrito
Call of Duty: Modern Warfare II	Call of Duty: Modern Warfare II	\$69.99	1	Añadir al carrito
Red Dead Redemption 2	Red Dead Redemption 2	\$59.99	1	Añadir al carrito
Grand Theft Auto V	Grand Theft Auto V	\$29.99	1	Añadir al carrito
Hogwarts Legacy	Hogwarts Legacy	\$59.99	1	Añadir al carrito
FIFA 24	FIFA 24	\$69.99	1	Añadir al carrito

Activar Windows

The screenshot shows a dark-themed confirmation page. At the top, a large green header reads "¡Pedido Confirmado!". Below the header, a message in white text says "Gracias por tu compra, tu pedido ha sido procesado exitosamente." A bold green text "Pedido ID: 2" follows, and below it, the total amount "Total a pagar: \$29.99" is displayed in white. At the bottom, a green button labeled "Volver a la tienda" (Return to store) is visible.

Pec Shoc - Carro de Compras

## Carrito de Compras

**Fortnite: Save the World**

Precio: \$29.99  
Cantidad: 1  
Total: \$29.99

Eliminar

**Total: \$29.99**

Proceder al Pago

**Super Mario Odyssey**

Precio: \$59.99  
Cantidad: 1  
Total: \$59.99

Eliminar

**Minecraft**

Precio: \$26.95  
Cantidad: 1  
Total: \$26.95

Eliminar

**Total: \$86.94**

Activar Windows  
Ve a Configuración para activar Windows

Proceder al Pago

Codigo de git hub

<https://github.com/AlanOsornio/Practica-Teja.git>

Codigo de Python anywhere

<http://Polo2007.pythonanywhere.com>