

# PYTHON

PROYECTO FINAL

Santiago Almazán Lalmolda



Santiago Almazán Lalmolda

RESPONSABLE DEL DESARROLLO DE LA APLICACIÓN

**Fecha:** XX, MES, AÑO



## MANUAL

El proyecto consta de las siguientes librerías/frameworks:

- Python 3.10.2
- Flask 2.2.2
- Jinja2 3.1.2
- SQLAlchemy 1.4.41
- Matplotlib 3.6.1
- Numpy 1.23.4
- SQLite3
- HTML5
- CSS

Instalación de los módulos:

1. Instalación de Python por medio de Python.org

The screenshot shows the Python.org homepage. The navigation bar includes links for Python, PSF, Docs, PyPI, Jobs, and Community. The main content area features a large Python logo and a brief introduction about Python being a programming language for quick work and system integration. On the left, there's a code snippet window showing a simple Python script. On the right, a prominent callout box for "Download for Windows" highlights Python 3.10.8, noting that it can run on Windows 7 or earlier. It also mentions that Python can be used on many other platforms like macOS and Linux. A note cautions against using Python 3.9+ on Windows 7 or earlier. At the bottom, there's a link to "Learn More".

2. Instalación de Flask desde el terminal de PyCharm por medio de pip install flask:



## Flask 2.2.2

```
pip install Flask
```



3. Instalación de Jinja2 desde el terminal de PyCharm por medio de pip install jinja2:

## Jinja2 3.1.2

```
pip install Jinja2
```



4. Instalación de SQLAlchemy desde el terminal de PyCharm por medio de pip install sqlalchemy:

## SQLAlchemy 1.4.42

```
pip install SQLAlchemy
```



5. Instalación de Matplotlib, nos instala también Numpy, desde el terminal de PyCharm por medio de pip install matplotlib:

# Installation

Install using pip:

```
pip install matplotlib
```



## ESTRUCTURA

La app web realizada en este proyecto final por mi es una página de ventas de productos informáticos en la cuál tengo distintas partes:

- página principal para la compra de los productos ofertados por cada proveedor,
- una página de carrito de compra para que los compradores realicen sus pedidos,
- una página login a la zona de proveedores,
- una página donde cada proveedor tiene sus propios productos y puede modificar tanto su información de proveedor como la de sus productos,
- sendas páginas para que se realice la modificación de la información del producto o información del proveedor que el mismo desee modificar,
- una página login para el administrador,
- una página donde el administrador podrá ver la información de todos los proveedores y sus productos y podrá modificar toda la información,
- sendas páginas para que se realice la modificación de la información del producto o información del proveedor que el administrador desee modificar,
- el main.py que será donde desarrollemos las funciones y rutas de la app web,
- el db.py donde creamos nuestra base de datos
- y el models.py donde creamos las tablas de la base de datos que sustentarán la información de nuestra app web.

## SOLICITADO

1. Nos piden tener inventariado todos sus productos y cuáles son sus cantidades en el almacén, de tal forma, que cuando el stock esté al 90% nos avise de pedir al proveedor.
2. En la aplicación web sería ideal tener dos tipos de acceso, uno para clientes y otro para proveedores. Además, un usuario administrador que tenga acceso a ambos.
3. Necesitaremos para nuestros clientes unas gráficas de ventas y para nuestros proveedores unas gráficas de compras. Para nosotros, tendremos unas gráficas comparativas, para saber lo que vendemos y los beneficios que sacamos. También se podrá buscar una alternativa para las gráficas, calculando unas estadísticas de ventas y compras y mostrando dichos resultados.



4. Todos los productos deben tener una descripción del producto, así como lo hay en el almacén, su precio, lugar donde se encuentra, etc. Aquí podéis tomaros licencias sobre la información extra que añadir, como número de referencia, colores...
5. Para los proveedores, debemos tener almacenados todos los datos de contacto (nombre de empresa, teléfono, dirección, cif...), facturación, precios de sus productos, porcentaje de descuento, IVA, etc.
6. Debemos elaborar la aplicación web de la forma más sencilla para el usuario y lo más práctica para nosotros en su manejo y obtención de datos importante para la empresa. Hay que tener en cuenta la Experiencia del Usuario la cual se caracteriza por sencillez, claridad, intuición.

## DESARROLLO

1. Lo primero que hago es crear la base de datos en mi db.py, para lo cuál importamos las partes de SQLAlchemy que necesitamos:

```
# Importamos las librerías necesarias
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker, declarative_base

# Creamos la base de datos 'productos'
engine = create_engine('sqlite:///database/registros.db', connect_args={'check_same_thread': False})
# 'connect_args' nos permite que la base de datos trabaje en segundo plano para evitar errores

# Creamos la sesión
Session = sessionmaker(bind=engine)
session = Session()

# Vinculamos la base de datos con nuestra clase
Base = declarative_base()
```

2. Creamos las tablas de nuestra base de datos, en el models.py, con toda la información que vamos a necesitar, para lo cual importamos las partes de SQLAlchemy que necesitamos y nuestra base de datos:

```
from sqlalchemy import Column, Integer, Float, String, Boolean, ForeignKey
import db # Este import lo realizamos para poder traernos Base y poder convertir las clases en tablas
from sqlalchemy.orm import relationship
```

Las tablas creadas son:

- Registro\_proveedores. Esta tabla tiene el objetivo de establecer la información que necesitaremos de cada proveedor.



```
# Clase para crear la tabla que lleva el registro de los proveedores que tenemos
class Registro_proveedores(db.Base): # Herencia, desde db.py, para poder convertirla en tablas
    __tablename__ = "proveedores" # Nombramos la segunda tabla para almacenar los proveedores
    __table_args__ = {'sqlite_autoincrement': True} # Aquí ponemos el autoincremento para id de referencia-primary key
    id = Column(Integer, primary_key=True) # Autoincremento para tener clasificados los proveedores
    proveedor = Column(String(200), nullable=False) # No puede ser None
    telefono = Column(Integer, nullable=False)
    direccion = Column(String(200), nullable=False)
    cif = Column(Integer, nullable=False)
    facturacion = Column(Float, nullable=False) # Apartado en el que se mostrarán las ganancias del proveedor
```

- Registro\_productos. En esta tabla tendremos toda la información referente a los productos que se venderán por parte de cada proveedor, así como la importación del id del proveedor para interrelacionar esta tabla con la anterior. Esta relación la utilizaremos para muchas cosas durante el desarrollo de la app web.

```
# Creamos esta clase para hacer la tabla con todos los productos a ser vendidos
class Registro_productos(db.Base): # Herencia, desde db.py, para poder convertirla en tablas
    __tablename__ = "productos" # Nombramos la primera tabla para almacenar los productos
    __table_args__ = {'sqlite_autoincrement': True} # Aquí ponemos el autoincremento para id de referencia-primary key
    id = Column(Integer, primary_key=True) # Autoincremento para tener clasificados los productos
    # Importamos el id de proveedores
    id_proveedor = Column(Integer, ForeignKey("proveedores.id", ondelete="CASCADE"), nullable=False)
    proveedor = relationship("Registro_proveedores") # Relación para obtener la información del proveedor
    nombre = Column(String(200), nullable=False) # No puede ser None
    tipo = Column(String(200), nullable=False)
    cantidad = Column(Integer, nullable=False)
    cantidad_requerida = Column(Integer, nullable=False)
    precio = Column(Float, nullable=False)
    iva = Column(Float, nullable=False)
    descuento = Column(Float, nullable=False)
    dimension = Column(String(200), nullable=False)
    lugar = Column(String(200), nullable=False)
```

- Facturacion. Esta tabla nos servirá para llevar un registro de cada pedido y de la persona, con su dirección, que lo realizó.

```
# Clase creada para llevar el registro de las ventas que se produzcan
class Facturacion(db.Base):
    __tablename__ = "facturacion" # Nombramos la segunda tabla para almacenar las ventas
    __table_args__ = {'sqlite_autoincrement': True} # Aquí ponemos el autoincremento para id de referencia-primary key
    id = Column(Integer, primary_key=True) # Autoincremento para tener clasificadas las compras
    nombre = Column(String(200), nullable=False)
    direccion = Column(String(200), nullable=False)
```

- Facturacion\_detalle. Esta tabla es un paso más en el tema de la facturación para obtener el producto comprado y la cantidad del mismo. Asimismo, importamos en esta tabla los id tanto de productos como de facturación para interrelacionarlos.



```
# Clase creada para llevar el detalle del registro de cada una de las ventas que se produzcan
class Facturacion_detalle(db.Base):
    __tablename__ = "facturacion_detalle" # Nombramos la tercera tabla para almacenar el detalle de las ventas
    __table_args__ = {'sqlite_autoincrement': True} # Aquí ponemos el autoincremento para id de referencia-primary key
    id = Column(Integer, primary_key=True) # Autoincremento para tener clasificadas las compras
    # Importamos el id de la venta
    id_factura = Column(Integer, ForeignKey("facturacion.id", ondelete="CASCADE"), nullable=False)
    factura = relationship("Facturacion") # Relación para obtener la información de la venta
    # Importamos el id de productos
    id_producto = Column(Integer, ForeignKey("productos.id", ondelete="CASCADE"), nullable=False)
    producto = relationship("Registro_productos") # Relación para obtener la información de los productos
    cantidad = Column(Integer, nullable=False)
    valor = Column(Float, nullable=False)
```

- Claves. En esta tabla lo que tenemos es el usuario y contraseña de cada uno de los proveedores y su perfil de acceso a cada una de las partes.

```
# Clase para crear en la base de datos una tabla con todos los usuarios y sus claves
class Claves(db.Base): # Herencia, desde db.py, para poder convertirla en tablas
    __tablename__ = "claves"
    __table_args__ = {'sqlite_autoincrement': True}
    id = Column(Integer, primary_key=True)
    usuario = Column(String(200), nullable=False) # No puede ser None
    clave = Column(String(200), nullable=False)
    perfil = Column(String(200), nullable=False)
```

- Cesta\_temporal. Esta última tabla nos sirve para poder almacenar, hasta que se realice la compra, los productos que el consumidor seleccione en nuestra página principal para comprarlos y tenerlos registrados en nuestra cesta de compra. Asimismo, relacionamos la tabla con el id del producto y el producto en si.

```
# Clase para crear la tabla temporal que nos guarde la cesta de la compra hasta que la compra se realice
class Cesta_temporal(db.Base): # Herencia, desde db.py, para poder convertirla en tablas
    __tablename__ = "cesta" # Nombramos la segunda tabla para almacenar los proveedores
    __table_args__ = {'sqlite_autoincrement': True} # Aquí ponemos el autoincremento para id de referencia-primary key
    id = Column(Integer, primary_key=True) # Autoincremento para tener clasificados los proveedores
    # Importamos el id del producto
    id_producto = Column(Integer, ForeignKey("productos.id", ondelete="CASCADE"), nullable=False)
    producto = relationship("Registro_productos") # Relación para obtener la información de los productos
    precio = Column(Float, nullable=False) # Precio incluido impuestos y descuento
```

### 3. Creamos la información de algunas de nuestras tablas:



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda\database\registros.db

1	1 Bionic	20bio20nic	proveedor
2	2 Forgeon	10For20ge30on	proveedor
3	3 MSI	15%MSI%15	proveedor
4	4 Tempest	25Tem15pest	proveedor
5	Xiaomi	30Xiao%10mi	proveedor
6	Administrador	3%Admin%5istrador	admin

DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda\database

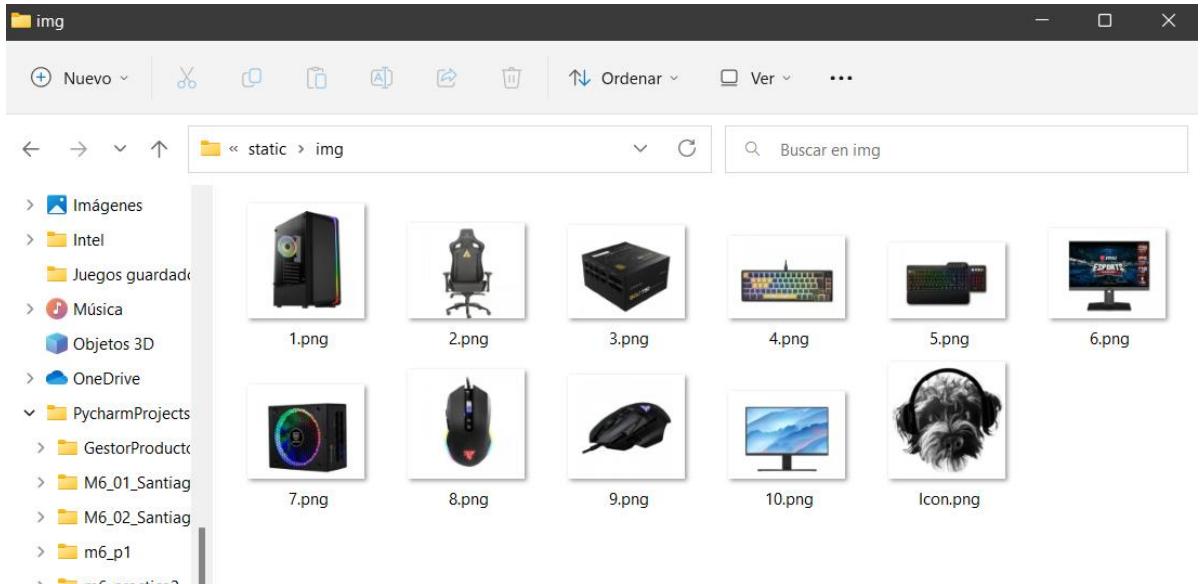
1	1 Bionic	986553169	Calle de los Abedules, 50	92465846	
2	2 Forgeon	916349513	Calle del Árbol, 34	267509373	
3	3 MSI	976842772	Avenida Roble, 5	608537248	
4	4 Tempest	956332624	Calle Secuoya, 13	932561783	
5	Xiaomi	936457594	Avenida Roble, 68	163489231	



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda\database\registros.db

	id	id_proveedor	nombre	tipo	cantidad	cantidad_requerid
1	1	1	Aerocool Bionic V2 RGB Cristal ...	Torre de ordenador	8	
2	2	2	Forgeon Acrux Leather Silla Gaming ...	Silla Gaming	8	
3	3	2	Forgeon Bolt PSU 750W 80+ Gold Ful...	Fuente de alimentación	11	
4	4	2	Forgeon Clutch Teclado Gaming RGB ...	Teclado	6	
5	5	3	MSI Max Teclado Gaming Mecánico ...	Teclado	4	
6	6	3	MSI Optix MAG245R2 23.8 pulgadas ...	Pantalla	4	
7	7	1	Bionic Sagitta RGB 650W 80 Plus Gol...	Fuente de alimentación	8	
8	8	4	Tempest X5S Strider RGB Ratón ...	Ratón	6	
9	9	4	Tempest X8 Keeper RGB Ratón Gami...	Ratón	4	
10	10	5	Xiaomi Mi Desktop Monitor 27 pulgad...	Pantalla	5	

4. Buscamos las imágenes de nuestros productos y creamos la imagen icono de nuestra página:



En el caso de las imágenes de nuestros productos, les ponemos números como nombre para que nos los conecte con el id del producto y luego en la app web nos muestre todo conjuntamente gracias a esa relación.

5. Buscamos los estilos que vamos a utilizar en nuestra app web:



- En Bootstrap CDN usamos el estilo Litera para la app web:

[Bootswatch](#) Themes ▾ Download ▾ Help Blog

# Litera

The medium is the message

Primary

Secondary

Success

Info

Warning

- En Google fonts escogemos la fuente Roboto para la app web:

Roboto

Christian Robertson

12 styles

Whereas  
recognition of the  
inherent dignity

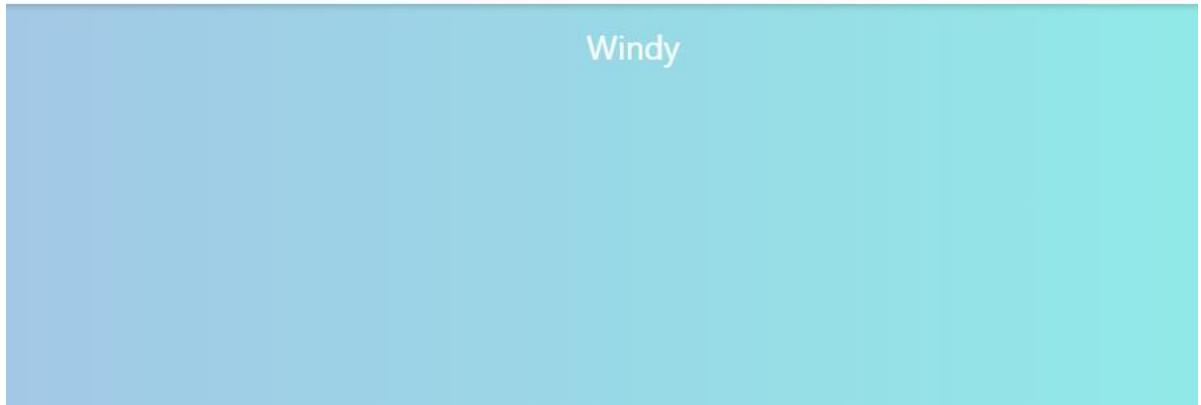
Los importamos en nuestro index.html y otros html que crearemos:



```
<head>
    <meta charset="UTF-8">
    <title>Suministros Informaticos</title>
    <!-- Vinculamos el html con el css -->
    <link rel="stylesheet" href="{{ url_for('static', filename='main.css') }}">
    <!-- Introducimos el Tema personalizado Sketchy desde BootstrapCDN -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootswatch@4.5.2/dist/litera/bootstrap.min.css" integrity=
    <!-- Fuente personalizada de Google Fonts: Roboto -->
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Roboto&display=swap" rel="stylesheet">
</head>
```

6. Buscamos el background de nuestra página en uiGradients.com y escogemos como fondo de nuestra página el background windy:

#acb6e5 → #86fde8



7. Configuramos nuestra página principal en nuestro index.html poniendo una barra de navegación, para mantener una estructura común en nuestra app web, y realizando un bucle for sobre nuestros productos desde la ruta principal en nuestro main.py:

- Imagen del resultado final de nuestra Página principal. En nuestra barra de navegación tenemos los accesos para volver a la página principal (en el nombre de Suministros Informáticos y en el Home), los accesos para proveedores y el administrador y la ruta directa para revisar el estado de nuestra cesta de compra



Aerocool Bionic V2 RGB Cristal Templado USB 3.0 Negra	Forgeon Acrux Leather Silla Gaming Negra	Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación
Torre de ordenador	Silla Gaming	Fuente de alimentación
Precio: 40.5 €	Precio: 149.95 €	Precio: 53.2 €
Cantidad: 8	Cantidad: 8	Cantidad: 11
<a href="#">Añadir a la cesta</a>	<a href="#">Añadir a la cesta</a>	<a href="#">Añadir a la cesta</a>

- Código de nuestra función home, con la lista\_de\_productos para iterar en el html y se vean los productos en cada una de las cards con su información correcta

```
# Creamos a la función 'home'
@app.route("/")
def home():
    # Variable "lista_de_productos" para que se nos de la información de todos los productos
    lista_de_productos = todos_los_productos()
    return render_template("index.html", productos=lista_de_productos)
```

- Código en nuestro index.html para la barra de navegación

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <div class="container-fluid">
        
        <a class="navbar-brand" href="/">Suministros Informaticos</a>
        <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarText" aria-cont
            <span class="navbar-toggler-icon"></span>
        </button>
        <div class="collapse navbar-collapse" id="navbarText">
            <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/">Home</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/login_proveedores">Acceso Proveedores</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/login_admin">Acceso Administrador</a>
                </li>
                <li class="nav-item">
                    <a class="nav-link active" aria-current="page" href="/carro_compra">Cesta de compra</a>
                </li>
            </ul>
            <span class="navbar-text" style="float:right"><strong>Página principal</strong></span>
        </div>
    </div>
</nav><br>
```



- Código en nuestro index.html con un bucle for, operado gracias a Jinja2, para mostrar cada producto con su información e imagen. Asimismo, con un botón para comprar los productos y que se añadan a nuestra cesta temporal, a la espera de ser comprados.

```
<main class="container p-4">
    <div class="row"> <!-- Indicamos que queremos que se organice la página en filas -->
        <!-- Hacemos nuestras etiquetas de productos -->
        <div class="row row-cols-1 row-cols-md-3 g-4">
            {%for producto in productos%}
                <div class="card mb-3">
                    <h6 class="card-header">{{producto.nombre}}</h6>
                    
                    <div class="card-body">
                        <p class="card-text">{{producto.tipo}}</p>
                    </div>
                    <ul class="list-group list-group-flush">
                        <li class="list-group-item"><strong>Precio:</strong> {{producto.precio}} €</li>
                        <li class="list-group-item"><strong>Cantidad:</strong> {{producto.cantidad}}</li>
                        {%if producto.descuento > 0%}
                            <li class="list-group-item"><strong>Descuento:</strong> {{producto.descuento}} %</li>
                            <li class="list-group-item"><strong>Precio descontado:</strong> {{producto.precio_descontado}} €</li>
                        {%endif%}
                    </ul>
                    <div class="card-body">
                        <form action="/compra" method="post">
                            <a type="submit" class="btn btn-primary" href="/carro_compra/{{producto.id}}">Añadir a la cesta</a>
                            <!--Añadimos el id del producto para guardarlo en una memoria temporal hasta que se realice la compra-->
                        </form>
                    </div>
                </div>
            {%endfor%}
        </div>
    </div>
```

8. Creación de nuestras páginas para que los proveedores se puedan registrar (login.html), acceder a su información como proveedores (proveedores.html) y modificarla:

**Registro de Proveedor**

Nombre Proveedor  
Forgeon

Contraseña  
\*\*\*\*\*

Acceder

- Para acceder y para registrarse como proveedor ponemos por separado las funciones como GET y POST, en nuestra función del main.py, para poder acceder



a esta página y luego poder iterar en nuestro html la información del proveedor que necesitemos. Asimismo, Creamos una excepción en caso de no registrarse de forma correcta el proveedor

```
# Creamos la ruta para ingresar a la pantalla de login del proveedor
@app.route("/login_proveedores", methods=["GET"])
def login():
    return render_template("login.html")

# Ruta para poder validar las credenciales y acceder como proveedor
@app.route("/login_proveedores", methods=["POST"])
def login_proveedores():
    try:
        proveedor=request.form.get("proveedor")
        clave=request.form.get("clave")
        if proveedor == "":
            raise Exception("Error, usuario vacío")
        if clave == "":
            raise Exception("Error, clave vacía")
        proveedor_a_mostrar = db.session.query(Claves).filter(Claves.usuario.like(proveedor)).filter(Claves.clave.like(clave))
        if proveedor_a_mostrar == None:
            raise Exception("Error, usuario o contraseña incorrectos")
        datos_proveedor = db.session.query(Registro_proveedores).filter(Registro_proveedores.proveedor==proveedor).first()
        todos_los_productos = info_productos(datos_proveedor.id)
        return render_template("proveedores.html", proveedor=proveedor, info_proveedor=datos_proveedor, lista_productos=todos_los_productos)
    except Exception as error:
        mensaje = {'error': error}
    return render_template("login.html", mensaje=mensaje)
```

- Funciones creadas para su reusabilidad durante el desarrollo del programa

```
# Función creada para poder obtener un listado de todos los productos de cada proveedor y mostrarlos
def info_productos(id_proveedor:int):
    return db.session.query(Registro_productos).filter(Registro_productos.id_proveedor==id_proveedor).all()

# Función para buscar los datos del producto por el id
def info_producto(id_producto:int):
    return db.session.query(Registro_productos).filter(Registro_productos.id==id_producto).first()

# Función creada para poder obtener un listado de todos los proveedores
def info_proveedor(id_proveedor):
    return db.session.query(Registro_proveedores).filter(Registro_proveedores.id==id_proveedor).first()

# Función para poder reutilizar el código y poder llamar a todos los productos que tenemos
def todos_los_productos():
    lista_de_productos = db.session.query(Registro_productos).all()
    return lista_de_productos
```

- Programación usada en login.html, con Jinja2, para realizar el registro del proveedor y acceder a la página de proveedores.html



```
<main class="form-signin w-25 m-auto">
    <form class="form-signin" action="/login_proveedores" method="POST">
        <h1 class="h3 mb-3 fw-normal">Registro de Proveedor</h1>

        <div class="form-floating">
            <input type="text" class="form-control" id="proveedor" name="proveedor" placeholder="Nombre proveedor">
            <label for="proveedor">Nombre Proveedor</label>
        </div>
        <div class="form-floating">
            <input type="password" class="form-control" id="clave" name="clave" placeholder="Contraseña">
            <label for="clave">Contraseña</label>
        </div>
        <a href="proveedores.html">
            <button class="btn is-success btn-success mt-2" type="submit">Acceder</button>
            {% if mensaje %}
                {{mensaje.error}}
            {% endif %}
            {%# Mostrar mensajes / errores si es que existen #}
            {% with mensajes_flash = get_flashed_messages() %}
            {% if mensajes_flash %}
                <div class="notification is-danger mt-2">
                    {% for mensaje in mensajes_flash %}
                        <li>{{ mensaje }}</li>
                    {% endfor %}
                </div>
            {% endif %}
            {% endwith %}
        </a>
    </form>

```

- Imagen del resultado del área de cada proveedor tras registrarse

The screenshot shows the 'Proveedor' (Supplier) section of the application. At the top, there's a navigation bar with a user icon, the text 'Suministros Informaticos', and 'Home Área del proveedor Forgeon'. Below this, a heading 'Información del proveedor:' is displayed. A table lists the supplier's details: 'Proveedor' (Forgeon), 'Teléfono' (916349513), 'Dirección' (Calle del Árbol, 34), and 'CIF' (267509373). An 'Editar' (Edit) button is located to the right of the table. Below this, another heading 'Información de los productos del proveedor:' is shown, followed by a table listing three products: 'Forgeon Acrux Leather Silla Gaming Negra', 'Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación', and 'Forgeon Clutch Teclado Gaming RGB 60% Switch Blue'. Each product row includes columns for 'Producto', 'Tipo', 'Cantidad Requerida', 'Precio', 'IVA', 'Descuento', 'Dimensión', 'Lugar', and 'Acciones' (with an 'Editar' button).

Proveedor	Teléfono	Dirección	CIF	Facturación	Acción
Forgeon	916349513	Calle del Árbol, 34	267509373		<a href="#">Editar</a>

Producto	Tipo	Cantidad	Requerida	Precio	IVA	Descuento	Dimensión	Lugar	Acciones
Forgeon Acrux Leather Silla Gaming Negra	Silla Gaming	8	8	149.95	10.0	0.0	120x50x50	Bodega	<a href="#">Editar</a>
Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación	Fuente de alimentación	11	12	53.2	0.0	0.0	10x10x5	Bodega	<a href="#">Editar</a>
Forgeon Clutch Teclado Gaming RGB 60% Switch Blue	Teclado	6	6	39.95	0.0	0.0	35x15x2	Tienda	<a href="#">Editar</a>

- Función, en nuestro main.py, para editar la información del proveedor, también con los métodos GET y POST puestos por separado para diferenciar sus usos



```
# Ruta y función para mostrar la información del proveedor
@app.route("/info_proveedor/<int:id>", methods=["GET"])
def proveedor(id):
    proveedor_a_mostrar = info_proveedor(id)
    return render_template("editar_proveedor.html", proveedor=proveedor_a_mostrar)

# Ruta y función para actualizar la información del proveedor
@app.route("/info_proveedor", methods=["POST"])
def proveedor_actualizado():
    # Establecemos los campos de la tabla a poder ser modificados en este apartado
    id = request.form.get("id")
    direccion = request.form.get("direccion")
    telefono = request.form.get("telefono")
    cif = request.form.get("cif")
    # Filtramos la tabla de proveedores por el id
    datos_proveedor = db.session.query(Registro_proveedores).filter(Registro_proveedores.id == id)
    # Modificamos la información del proveedor en la base de datos
    datos_proveedor.update(dict(telefono=telefono, direccion=direccion, cif=cif))
    db.session.commit() # Guardamos los cambios realizados
    # Leemos nuevamente la información modificada
    proveedor = datos_proveedor.first()
    todos_los_productos = info_productos(proveedor.id)
    return render_template("proveedores.html", info_proveedor=proveedor, lista_productos=todos_los_productos)
```

- Imagen de la página editar\_proveedor.html y código utilizado para mostrarla y modificarla

#### **Edición del proveedor:**

Proveedor:

Forgeon

Teléfono:

916349513

Dirección:

Calle del Árbol, 34

CIF:

267509373

**Guardar cambios**

La única información que no se puede modificar en el nombre del proveedor que, como vemos a continuación en el código, se ha deshabilitado la posibilidad.



```
<div class="container">
<section>
    <!--Lista de los datos del proveedor-->
    <h4>Edición del proveedor:</h4><br>
    <!--Formulario para modificar la información del proveedor-->
    <form action="/info_proveedor" method="post">
        <input value="{{proveedor.id}}" type="hidden" class="form-control" id="id" name="id">
        <div class="mb-3">
            <label for="proveedor" class="form-label">Proveedor:</label>
            <input value="{{proveedor.proveedor}}" disabled="disabled" type="text" class="form-control" id="proveedor" name="proveedor">
        </div>
        <div class="mb-3">
            <label for="telefono" class="form-label">Teléfono:</label>
            <input value="{{proveedor.telefono}}" type="text" class="form-control" id="telefono" name="telefono">
        </div>
        <div class="mb-3">
            <label for="direccion" class="form-label">Dirección:</label>
            <input value="{{proveedor.direccion}}" type="text" class="form-control" id="direccion" name="direccion">
        </div>
        <div class="mb-3">
            <label for="cif" class="form-label">CIF:</label>
            <input value="{{proveedor.cif}}" type="text" class="form-control" id="cif" name="cif">
        </div>
        <button type="submit" class="btn btn-primary">Guardar cambios</button>
    </form>
</section>
</div>
```

Muestra de que funcionan los cambios al modificar la dirección:

#### Información del proveedor:

Proveedor	Teléfono	Dirección	CIF	Facturación	Acción
Forgeon	916349513	Calle Arboleda, 52	267509373		<a href="#">Editar</a>

- Función, en nuestro main.py, para editar la información de los productos del proveedor registrado, también con los métodos GET y POST puestos por separado para diferenciar sus usos

```
# Ruta y función para mostrar la información del producto
@app.route("/info_producto/", methods=["GET"])
def producto(id):
    producto_a_mostrar = info_producto(id)
    return render_template("editar_productos.html", producto=producto_a_mostrar)
```



```

@app.route("/info_producto", methods=["POST"])
def producto_actualizado():
    # Establecemos los campos de la tabla a poder ser modificados en este apartado
    id = request.form.get("id")
    nombre = request.form.get("nombre")
    tipo = request.form.get("tipo")
    cantidad = request.form.get("cantidad")
    cantidad_requerida = request.form.get("cantidad_requerida")
    precio = request.form.get("precio")
    iva = request.form.get("iva")
    descuento = request.form.get("descuento")
    dimension = request.form.get("dimension")
    lugar = request.form.get("lugar")
    # Filtramos la tabla de proveedores por el id
    datos_producto = db.session.query(Registro_productos).filter(Registro_productos.id == id)
    datos_producto.update(dict(nombre=nombre,
                               tipo=tipo,
                               cantidad=cantidad,
                               cantidad_requerida=cantidad_requerida,
                               precio=precio,
                               iva=iva,
                               descuento=descuento,
                               dimension=dimension,
                               lugar=lugar)) # Modificamos la información del proveedor en la base de datos
    db.session.commit() # Guardamos los cambios realizados
    # Leemos nuevamente la información modificada
    producto=datos_producto.first()
    todos_los_productos = info_productos(producto.id_proveedor)
    proveedor=info_proveedor(producto.id_proveedor)
    return render_template("proveedores.html", info_proveedor=proveedor, lista_productos=todos_los_productos)

```

- Imagen de la página editar\_productos.html y código utilizado para mostrarla y modificarla

### Edición del producto:

Producto:

Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación

Tipo:

Fuente de alimentación

Cantidad:

11



Cantidad Requerida:

Precio:

IVA:

Descuento:

Dimensiones del producto:

Lugar en el que se encuentra el producto:

[Guardar cambios](#)

**Código HTML:**

```
<div class="container">
    <section>
        <!--Lista de los datos del proveedor-->
        <h4>Edición del producto:</h4><br>
        <!--Formulario para modificar la información del proveedor-->
        <form action="/info_producto" method="post">
            <input value="{{producto.id}}" type="hidden" class="form-control" id="id" name="id">
            <div class="mb-3">
                <label for="nombre" class="form-label">Producto:</label>
                <input value="{{producto.nombre}}" type="text" class="form-control" id="nombre" name="nombre">
            </div>
            <div class="mb-3">
                <label for="tipo" class="form-label">Tipo:</label>
                <input value="{{producto.tipo}}" type="text" class="form-control" id="tipo" name="tipo">
            </div>
            <div class="mb-3">
                <label for="cantidad" class="form-label">Cantidad:</label>
                <input value="{{producto.cantidad}}" type="text" class="form-control" id="cantidad" name="cantidad">
            </div>
            <div class="mb-3">
                <label for="cantidad_requerida" class="form-label">Cantidad Requerida:</label>
                <input value="{{producto.cantidad_requerida}}" type="text" class="form-control" id="cantidad_requerida" name="cantidad_requerida">
            </div>
            <div class="mb-3">
                <label for="precio" class="form-label">Precio:</label>
                <input value="{{producto.precio}}" type="text" class="form-control" id="precio" name="precio">
            </div>
        </form>
    </section>
</div>
```



```
</div>
<div class="mb-3">
    <label for="iva" class="form-label">IVA:</label>
    <input value="{{producto.iva}}" type="text" class="form-control" id="iva" name="iva">
</div>
<div class="mb-3">
    <label for="descuento" class="form-label">Descuento:</label>
    <input value="{{producto.descuento}}" type="text" class="form-control" id="descuento" name="descuento">
</div>
<div class="mb-3">
    <label for="dimension" class="form-label">Dimensiones del producto:</label>
    <input value="{{producto.dimension}}" type="text" class="form-control" id="dimension" name="dimension">
</div>
<div class="mb-3">
    <label for="lugar" class="form-label">Lugar en el que se encuentra el producto:</label>
    <input value="{{producto.lugar}}" type="text" class="form-control" id="lugar" name="lugar">
</div>
<button type="submit" class="btn btn-primary">Guardar cambios</button>
</form>
ction>
```

Funciones y rutas del main.py para operar y guardar los cambios, usando métodos GET y POST para diferenciar sus usos:

```
# Ruta y función para mostrar la información del producto
@app.route("/info_producto/<int:id>", methods=["GET"])
def producto(id):
    producto_a_mostrar = info_producto(id)
    return render_template("editar_productos.html", producto=producto_a_mostrar)
```



```

@app.route("/info_producto", methods=["POST"])
def producto_actualizado():
    # Establecemos los campos de la tabla a poder ser modificados en este apartado
    id = request.form.get("id")
    nombre = request.form.get("nombre")
    tipo = request.form.get("tipo")
    cantidad = request.form.get("cantidad")
    cantidad_requerida = request.form.get("cantidad_requerida")
    precio = request.form.get("precio")
    iva = request.form.get("iva")
    descuento = request.form.get("descuento")
    dimension = request.form.get("dimension")
    lugar = request.form.get("lugar")
    # Filtramos la tabla de proveedores por el id
    datos_producto = db.session.query(Registro_productos).filter(Registro_productos.id == id)
    datos_producto.update(dict(nombre=nombre,
                               tipo=tipo,
                               cantidad=cantidad,
                               cantidad_requerida=cantidad_requerida,
                               precio=precio,
                               iva=iva,
                               descuento=descuento,
                               dimension=dimension,
                               lugar=lugar)) # Modificamos la información del proveedor en la base de datos
    db.session.commit() # Guardamos los cambios realizados
    # Leemos nuevamente la información modificada
    producto=datos_producto.first()
    todos_los_productos = info_productos(producto.id_proveedor)
    proveedor=info_proveedor(producto.id_proveedor)
    return render_template("proveedores.html", info_proveedor=proveedor, lista_productos=todos_los_productos)

```

Muestra de que funcionan los cambios al modificar la cantidad:

Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación	Fuente de alimentación	12	12	53.2	0.0	0.0	10x10x5	Bodega	<a href="#">Editar</a>
--	------------------------	----	----	------	-----	-----	---------	--------	------------------------

9. Creación de la cesta temporal y su página (carro\_compra.html) para poder realizar la compra:

- Acceso tanto desde la barra de nuestro navegador como desde el botón de añadir a la cesta de cada producto

Imagen y código de acceso desde la barra navegador de nuestra app web:





Código HTML de la página carro\_compra.html, uso de Jinja2 para obtener los datos de cada producto añadido a la cesta, el número de productos añadidos a la misma y la cantidad total a pagar por los mismos en caso de compra. Cuenta con un botón para volver al home y seguir comprando:

```
<main class="container p-4">
    <div class="row g-5">
        <div class="col-md-5 col-lg-4 order-md-last">
            <h4 class="d-flex justify-content-between align-items-center mb-3">
                <span class="text-primary">Tus productos</span>
                <!--Con cesta|length obtenemos por medio de Jinja2 la longitud para ver cuantos productos hay en la cesta-->
                <span class="badge bg-primary rounded-pill">{{cesta|length}}</span>
            </h4>
            <ul class="list-group mb-3">
                {%for producto in cesta%}
                    <li class="list-group-item d-flex justify-content-between lh-sm">
                        <div>
                            <h6 class="my-0">{{producto.producto.nombre}}</h6>
                            <small class="text-muted">{{producto.producto.tipo}}</small>
                        </div>
                        <span class="text-muted">{{producto.producto.precio_descontado}} €</span>
                    </li>
                {%endfor%}
                <li class="list-group-item d-flex justify-content-between">
                    <span>Total (€)</span>
                    <strong>{{total_cesta}} €</strong>
                </li>
                <hr class="my-4">
                <a class="w-100 btn btn-danger btn-lg" type="submit" href="/eliminar_carro">Eliminar carrito</a>
            </ul>
        </div>
        <div class="col-md-7 col-lg-8">
            <h4 class="mb-3">Información del comprador</h4>
            {%if mensaje%}
                <p>{{mensaje.error}}</p>
            {%endif%}
            <form class="needs-validation" action="/compra" method="post">
                <div class="row g-3">
                    <div class="col-12">
                        <label for="nombre" class="form-label">Nombre</label>
                        <input name="nombre" type="text" class="form-control" id="nombre" placeholder="Ingrese su nombre y apellidos" required>
                    </div>
                    <div class="col-12">
                        <label for="direccion" class="form-label">Dirección</label>
                        <input name="direccion" type="text" class="form-control" id="direccion" placeholder="Ingrese su dirección" required>
                    </div>
                </div>
                <hr class="my-4">
                <button class="w-100 btn btn-success btn-lg" type="submit">Realizar compra</button>
                <hr class="my-4">
                <a class="w-100 btn btn-primary btn-lg" href="/">Continuar comprando</a>
            </form>
        </div>
    </div>
</main>
```

Función y ruta en nuestro main.py para configurar el carro y que se nos muestre al hacer click en el mismo desde la barra de navegación:



```
# Ruta y función para que el acceso al carrito de la compra, donde se muestra lo que tenemos en el carrito
@app.route("/carro_compra", methods=["GET"])
def carro_compra():
    cesta = db.session.query(Cesta_temporal).all() # mostramos los productos en la cesta temporal
    total_precio = 0
    for datos in cesta:
        total_precio += datos.producto.precio_descontado
    total_precio = round(total_precio, 2)
    return render_template("carro_compra.html", cesta=cesta, total_cesta=total_precio)
```

Imagen de acceso por medio del botón de "Añadir a la cesta":

Código en el main.py para añadir los productos desde el botón en el index.html, usamos el precio ya con descuentos de nuestra tabla para que se sumen en el total los precios definitivos y redondeamos a 2 decimales:

```
# Ruta y función para crear la cesta temporal, en la que se inserta el producto
@app.route("/carro_compra/<int:id>", methods=["GET"])
def agregar_cesta(id):
    producto=info_producto(id) # Obtenemos los datos del producto
    precio=producto.precio_descontado # Obtenemos el precio del producto según su id
    cesta = Cesta_temporal(id_producto=id, precio=precio)
    db.session.add(cesta) # Añadimos el objeto cesta a la cesta temporal a la espera de que se realice la compra
    db.session.commit() # Guardamos
    cesta=db.session.query(Cesta_temporal).all() # mostramos los productos en la cesta temporal
    total_precio=0
    for datos in cesta:
        total_precio += datos.producto.precio_descontado
    total_precio = round(total_precio, 2)
    return render_template("carro_compra.html", cesta=cesta, total_cesta=total_precio)
```

Código e imagen demostrativa de que la cesta se vacía cuando se le da al botón de "eliminar carrito":



```
# Función para eliminar todos los elementos de la tabla "cesta" y poder reutilizarla
def eliminar_toda_la_cesta():
    db.session.query(Cesta_temporal).delete() # Eliminamos los elementos
    db.session.commit() # Guardamos los cambios

# Función y Ruta para eliminar todos los elementos del carro de compra cuando le damos al botón de eliminar
@app.route("/eliminar_carro", methods=["GET"])
def eliminar_cesta():
    eliminar_toda_la_cesta()
    return home() # Nos devuelve a nuestro "home"
```



Código que ejecuta la opción del botón de realizar compra. Cuenta con funciones para eliminar los productos que son comprados de la cantidad, la adición de la información en las tablas de facturación y facturación\_detalle y excepción en caso de no haber la cantidad de productos solicitados. Asimismo, operaciones para obtener el valor de los productos sin iva y sin descuento, de haberlo, para el ítem valor de la tabla "facturación\_detalle":

```
# Función y ruta para comprar todos los productos y eliminarlos de la cesta temporal
@app.route("/compra", methods=["POST"])
def compra():
    try:
        # Recuperamos los datos de nombre y dirección del formulario en nuestro HTML, con la etiqueta "name"
        nombre = request.form.get("nombre")
        dirección = request.form.get("dirección")
        facturación = Facturación(nombre=nombre, dirección=dirección)
        db.session.add(facturación) # Añadimos la información
        db.session.flush() # Refrescamos los cambios producidos en el objeto
        # Traemos los productos y los agrupamos para poder contarlos en la tabla de Facturación_detalle
        grupo_productos = db.session.query(Cesta_temporal.id_producto, Cesta_temporal.precio, func.count(Cesta_temporal.id_producto))
        id_facturación = facturación.id
        for productos in grupo_productos:
            # Realizamos la resta de productos que se compran
            datos_producto = db.session.query(Registro_productos).filter(Registro_productos.id == productos[0]).first()
            total_stock = datos_producto.cantidad
            iva = datos_producto.iva
            valor_iva = (productos[1] * iva) / 100
            valor_sin_iva = round(productos[1] - valor_iva, 2)
```



```

if total_stock < productos[2]:
    raise Exception("Error, no hay stock suficiente")
else:
    cantidad_actualizada=total_stock - productos[2]
    db.session.query(Registro_productos).filter(Registro_productos.id==productos[0]).update(dict(cantidad=cantidad_actualizada))
# Creamos la variable "detalle" para completar los campos de la tabla "Facturacion_detalle"
detalle=Facturacion_detalle(id_factura=id_facturacion, id_producto=productos[0], valor=valor_sin_iva, cantidad=cantidad_actualizada)
db.session.add(detalle) # Añadimos a la tabla la información
db.session.commit() # Guardamos los cambios
eliminar_toda_la_cesta()
return home() # Nos devuelve a nuestro "home"
except Exception as error:
    mensaje = {'error': error}
    return render_template("carro_compra.html", mensaje=mensaje)

```

Imagen de los cambios en la base de datos antes y tras realizarse compras, incluyendo la cesta temporal:

### Información del comprador

Nombre  
Diego Almagro

Dirección  
Avenida real, 54

[Realizar compra](#)

[Continuar comprando](#)

### Tus productos

Forgeon Acrux Leather Silla Gaming Negra	134.95 €
Silla Gaming	
Forgeon Clutch Teclado Gaming RGB 60% Switch Blue	39.95 €
Teclado	
Aerocool Bionic V2 RGB Cristal Templado USB 3.0 Negra	40.5 €
Torre de ordenador	
Total (€)	<b>215.4 €</b>

[Eliminar carrito](#)



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Al

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshacer c

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: cesta

	id	id_producto	precio
1	24	2	134.95
2	25	4	39.95
3	26	1	40.5



The screenshot shows two separate windows of DB Browser for SQLite. The top window displays the 'facturacion' table with columns: id, nombre, and direccion. The bottom window displays the 'facturacion\_detalle' table with columns: id, id\_factura, id\_producto, cantidad, and valor.

facturacion		
id	nombre	direccion
Filtro	Filtro	Filtro

facturacion_detalle				
id	id_factura	id_producto	cantidad	valor
Filtro	Filtro	Filtro	Filtro	Filtro

Le damos al botón de "Realizar compra" y mostramos todo tras la compra:

The screenshot shows the 'facturacion' table with one row added after a purchase was made. The new row contains: id (1), nombre (4), and direccion (Avenida real, 54).

facturacion		
id	nombre	direccion
Filtro	Filtro	Filtro
1	4	Diego Almagro Avenida real, 54

En la siguiente tabla mostramos el valor ya sin iva y sin descuentos, de haberlos, para obtener el beneficio que obtiene el proveedor tras la compra del artículo y poder obtener la facturación total de cada proveedor:



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios:

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: facturacion\_detalle

	id	id_factura	id_producto	cantidad	valor
1	7	4	1	1	36.45
2	8	4	2	1	121...
3	9	4	4	1	35.96

Cesta vacía tras ser realizada la compra, pues se borra con cada compra:

DB Browser for SQLite - C:\Users\si

Archivo Editar Ver Herramientas Ayu

Nueva base de datos Abrir base

Estructura Hoja de datos Editar pr

Tabla: cesta

	id	id_producto	precio
	Filtro	Filtro	Filtro

Productos restados de cantidad tras la compra:



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto_Final_Santiago_Almazán_Lalmolda\database\registros.db					
Archivo Editar Ver Herramientas Ayuda					
Estructura		Hoja de datos		Editar pragmas Ejecutar SQL	
Tabla: productos					
proveedor	nombre	tipo	cantidad	cantidad_requerida	
	Filtro	Filtro	Filtro	Filtro	
1	Aerocool Bionic V2 RGB Cristal ...	Torre de ordenador	9	10	
2	Forgeon Acrux Leather Silla Gaming ...	Silla Gaming	7	8	
3	Forgeon Bolt PSU 750W 80+ Gold Ful...	Fuente de alimentación	12	12	
4	Forgeon Clutch Teclado Gaming RGB ...	Teclado	5	6	
5	MSI Max Teclado Gaming Mecánico ...	Teclado	4	4	
6	MSI Optix MAG245R2 23.8 pulgadas ...	Pantalla	4	4	
7	Bionic Sagitta RGB 650W 80 Plus Gol...	Fuente de alimentación	8	8	
8	Tempest X5S Strider RGB Ratón ...	Ratón	6	6	
9	Tempest X8 Keeper RGB Ratón Gami...	Ratón	4	4	
10	Xiaomi Mi Desktop Monitor 27 pulgad...	Pantalla	5	5	

#### 10. Acceso del administrador y páginas para que pueda ver y modificar todos los parámetros:

- Código para que el administrador se registre y acceda a su área, tanto en el main.py, con el uso de los métodos GET y POST para diferenciar las acciones que realizan, como en login\_admin.html

En nuestro método POST, cuando se acceda correctamente, enviamos toda la información que se requerirá en nuestro admin.html para mostrar toda la información:

```
# Ruta y función para que el administrador se loguee y tenga acceso a toda la información
@app.route("/login_admin", methods=["GET"])
def login_admin():
    return render_template("login_admin.html")

# Ruta para poder validar las credenciales y acceder como proveedor
@app.route("/login_admin", methods=["POST"])
def login_administrador():
    try:
        administrador=request.form.get("administrador")
        clave=request.form.get("clave")
        if administrador == "":
            raise Exception("Error, usuario vacío")
        if clave == "":
            raise Exception("Error, clave vacía")
        administrador_a_mostrar = db.session.query(Claves).filter(Claves.usuario.like(administrador)).filter(Claves.clave.like(clave))
        if administrador_a_mostrar == None:
            raise Exception("Error, usuario o contraseña incorrectos")
        datos_proveedor = db.session.query(Registro_proveedores).all()
        todos_los_productos = db.session.query(Registro_productos).all()
        return render_template("admin.html", administrador=administrador, administrador_a_mostrar=administrador_a_mostrar)
    except Exception as error:
        mensaje = {'error': error}
        return render_template("login_admin.html", mensaje=mensaje)
```

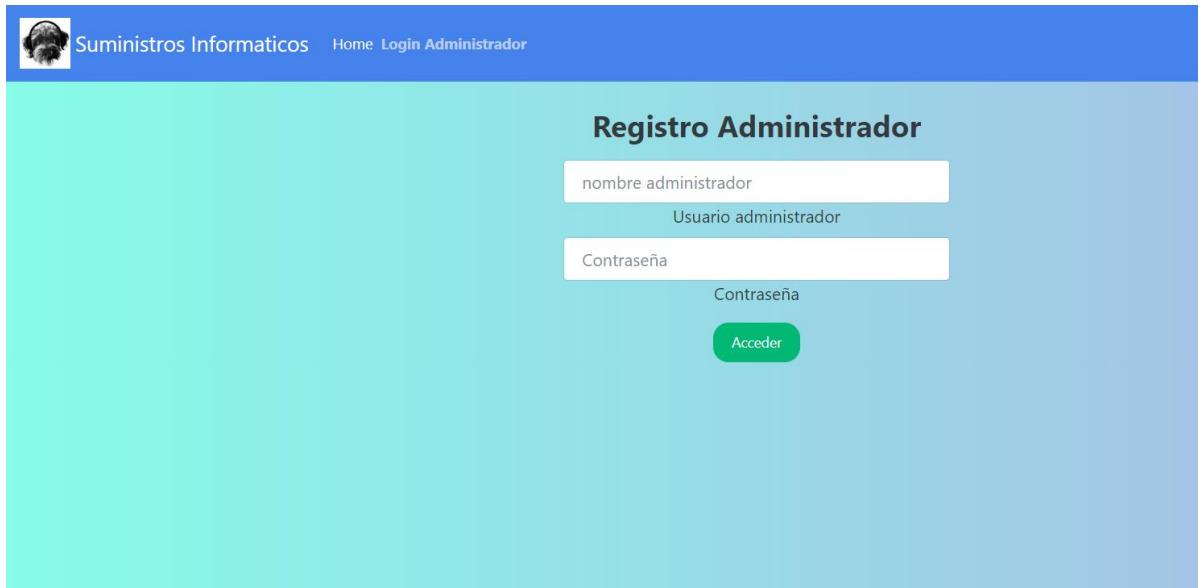


Código HTML para que el administrador se registre. Cuenta con el mismo mensaje de error que en el acceso de proveedores para el caso de no ser introducidos de forma correcta los campos:

```
<form class="form--signin" action="/login_admin" method="POST">
    <h1 class="h3 mb-3 fw-normal">Registro Administrador</h1>

    <div class="form-floating">
        <input type="text" class="form-control" id="administrador" name="administrador" placeholder="nombre administrador" required="required" value=""/>
        <label for="administrador">Usuario administrador</label>
    </div>
    <div class="form-floating">
        <input type="password" class="form-control" id="clave" name="clave" placeholder="Contraseña" required="required" value=""/>
        <label for="clave">Contraseña</label>
    </div>
    <a href="admin.html">
        <button class="btn is-success btn-success mt-2" type="submit">Acceder</button>
        {% if mensaje %}
            {{mensaje.error}}
        {% endif %}
        {%# Mostrar mensajes / errores si es que existen #}
        {% with mensajes_flash = get_flashed_messages() %}
            {% if mensajes_flash %}
                <div class="notification is-danger mt-2">
                    {% for mensaje in mensajes_flash %}
                        <li>{{ mensaje }}</li>
                    {% endfor %}
                </div>
            {% endif %}
            {% endwith %}
        </a>
    </form>
```

Imagen de la página de registro en nuestra app web:





- Al acceder, el administrador en su página puede visualizar la información de todos los proveedores así como todos los productos que se encuentran a la venta y editarlos si así lo requiere:

Proveedor	Teléfono	Dirección	CIF	Facturación	Acción
Bionic	986553169	Calle de los Abedules, 50	92465846	72.9 €	<button>Editar</button>
Forgeon	916349630	Calle Arboleda, 52	267509373	35.96 €	<button>Editar</button>
MSI	976842772	Avenida Roble, 5	608537248	0.0 €	<button>Editar</button>
Tempest	956332624	Calle Secuoya, 13	932561783	53.06 €	<button>Editar</button>
Xiaomi	936457594	Avenida Roble, 68	163489231	107.95 €	<button>Editar</button>

Proveedor	Producto	Tipo	Cantidad Requerida	Precio	IVA	Descuento	Dimensión	Lugar	Acciones
Bionic	AeroCool Bionic V2 RGB Cristal Templado USB 3.0 Negra	Torre de ordenador	8*	40.5	10.0	0.0	30x30x15	Bodega	<button>Editar</button>
Forgeon	Forgeon Acrux Leather Silla Gaming Negra	Silla Gaming	7*	149.95	10.0	10.0	120x50x50	Bodega	<button>Editar</button>
Forgeon	Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación	Fuente de alimentación	12	53.2	10.0	0.0	10x10x5	Bodega	<button>Editar</button>
Forgeon	Forgeon Clutch Teclado Gaming RGB 60% Switch Blue	Teclado	5*	39.95	10.0	0.0	35x15x2	Tienda	<button>Editar</button>
MSI	MSI Max Teclado Gaming Mecánico Custom Media Dark RGB Modular Cherry	Teclado	4	79.95	10.0	0.0	30x20x2	Bodega	<button>Editar</button>

Código en admin.html, este utiliza el método POST explicado anteriormente para que podamos usar Jinja2 y mostrar todos los campos que requerimos cuando iteremos con un bucle for tanto para mostrar los proveedores y su información como para los productos y su información:



```

<h4>Información de los proveedores:</h4><br>
<table class="table table-hover">
    <thead>
        <tr>
            <th scope="col">Proveedor</th>
            <th scope="col">Teléfono</th>
            <th scope="col">Dirección</th>
            <th scope="col">CIF</th>
            <th scope="col">Facturación</th>
            <th scope="col">Acción</th>
        </tr>
    </thead>
    {%for proveedor in info_proveedor%}
    <tbody>
        <tr>
            <td scope="row">{{proveedor.proveedor}}</td>
            <td>{{proveedor.telefono}}</td>
            <td>{{proveedor.direccion}}</td>
            <td>{{proveedor.cif}}</td>
            <td>{{proveedor.facturacion}}</td>
            <!--Botón para editar la información-->
            <td>
                <a class="btn btn-sm btn-primary mt-2" href="/info_proveedores_admin/{{proveedor.id}}" style="text-decoration: none; color: inherit;">Editar</a>
            </td>
        </tr>
    {%endfor%}
    <h4>Información de los productos de los proveedores:</h4><br>
    <table class="table table-hover">
        <thead>
            <tr>
                <th scope="col">Proveedor</th>
                <th scope="col">Producto</th>
                <th scope="col">Tipo</th>
                <th scope="col">Cantidad</th>
                <th scope="col">Cantidad Requerida</th>
                <th scope="col">Precio</th>
                <th scope="col">IVA</th>
                <th scope="col">Descuento</th>
                <th scope="col">Dimensión</th>
                <th scope="col">Lugar</th>
                <th scope="col">Acciones</th>
            </tr>
        </thead>
        <tbody>
            {% for productos in lista_productos %}
            <tr>
                <td scope="row">{{productos.proveedor.proveedor}}</td>
                <td>{{productos.nombre}}</td>
                <td>{{productos.tipo}}</td>
                {% if (productos.cantidad*100)/productos.cantidad_requerida < 90 %}
                <td><span class="badge rounded-pill bg-danger">{{productos.cantidad}}*</span></td>
                {%else%}
                <td>{{productos.cantidad}}</td>
                {%endif%}
            
```



```

        <!--Hacer mensaje para reponer productos si baja del 90%-->
        <td>{{productos.cantidad_requerida}}</td>
        <td>{{productos.precio}}</td>
        <td>{{productos.iva}}</td>
        <td>{{productos.descuento}}</td>
        <td>{{productos.dimension}}</td>
        <td>{{productos.lugar}}</td>
        <!--Botón para editar la información-->
        <td>
            <a class="btn btn-sm btn-primary mt-2" href="/info_productos_admin/{{productos.id}}" style="...>
                Editar
            </a>
        </td>
    </tr>
    {% endfor %}
</tbody>
</table><br>
<p>* El proveedor debe reponer, la cantidad es menor al 90% de la cantidad requerida</p><br>

```

El párrafo del final es un recordatorio para el administrador de la alerta que también le está saliendo al proveedor en caso de tener menos del 90% de cantidad del producto de la cantidad que se requiere que haya.

- Edición de la información del proveedor por el administrador

La visualización es la misma que cuando accedemos desde proveedor.html:

**Edición del proveedor:**

Proveedor:  
MSI

Teléfono:  
976842772

Dirección:  
Avenida Roble, 5

CIF:  
608537248

**Guardar cambios**

En el código de la función, proveedor\_admin(), y en el del editar\_proveedor\_admin.html cambian principalmente 2 cosas con respecto a las funciones y html que teníamos para los proveedores. Estas son que ahora debemos de filtrar por el id del proveedor no desde el login si no desde dentro de la función y que al guardar los cambios nos devuelva a la



página admin.html. El código modificado, también con los métodos GET y POST para diferenciar sus funcionalidades, de ambos es el siguiente:

```
# Ruta y función para mostrar la información de los proveedores al administrador
@app.route("/info_proveedores_admin/<int:id>", methods=["GET"])
def proveedor_admin(id):
    proveedor_a_mostrar = info_proveedor(id)
    return render_template("editar_proveedor_admin.html", proveedor=proveedor_a_mostrar)

# Ruta y función para actualizar la información de los proveedores por el administrador
@app.route("/info_proveedores_admin", methods=["POST"])
def proveedor_actualizado_admin():
    # Establecemos los campos de la tabla a poder ser modificados en este apartado
    id = request.form.get("id")
    direccion = request.form.get("direccion")
    telefono = request.form.get("telefono")
    cif = request.form.get("cif")
    # Filtramos la tabla de proveedores por el id
    datos_proveedor = db.session.query(Registro_proveedores).filter(Registro_proveedores.id == id)
    # Modificamos la información del proveedor en la base de datos
    datos_proveedor.update(dict(telefono=telefono, direccion=direccion, cif=cif))
    db.session.commit() # Guardamos los cambios realizados
    # Leemos nuevamente la información modificada
    proveedor = db.session.query(Registro_proveedores).all()
    todos_los_productos = db.session.query(Registro_productos).all()
    return render_template("admin.html", info_proveedor=proveedor, lista_productos=todos_los_productos)

<!--Lista de los datos del proveedor-->
<h4>Edición del proveedor:</h4><br>
<!--Formulario para modificar la información del proveedor-->
<form action="/info_proveedores_admin" method="post">
    <input value="{{proveedor.id}}" type="hidden" class="form-control" id="id" name="id">
    <div class="mb-3">
        <label for="proveedor" class="form-label">Proveedor:</label>
        <input value="{{proveedor.proveedor}}" disabled="disabled" type="text" class="form-control" id="proveedor" name="proveedor">
    </div>
    <div class="mb-3">
        <label for="telefono" class="form-label">Teléfono:</label>
        <input value="{{proveedor.telefono}}" type="text" class="form-control" id="telefono" name="telefono">
    </div>
    <div class="mb-3">
        <label for="direccion" class="form-label">Dirección:</label>
        <input value="{{proveedor.direccion}}" type="text" class="form-control" id="direccion" name="direccion">
    </div>
    <div class="mb-3">
        <label for="cif" class="form-label">CIF:</label>
        <input value="{{proveedor.cif}}" type="text" class="form-control" id="cif" name="cif">
    </div>
    <button type="submit" class="btn btn-primary">Guardar cambios</button>
</form>
<hr class="my-4">
```

- Muestra de que al guardar los cambios estos se modifican:

Base de datos antes de los cambios:



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda

proveedores						
	id	proveedor	telefono	direccion	cif	facturacion
1	1	Bionic	986553169	Calle de los Abedules, 50	92465846	72.9
2	2	Forgeon	916349630	Calle Arboleda, 52	267509373	35.96
3	3	MSI	976842772	Avenida Roble, 5	608537248	0.0
4	4	Tempest	956332624	Calle Secuoya, 13	932561783	53.06
5	5	Xiaomi	936457594	Avenida Roble, 68	163489231	107.95

Base de datos tras cambiar la dirección y el CIF, con imagen de la modificación en admin.html:

DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda

proveedores						
	id	proveedor	telefono	direccion	cif	facturacion
1	1	Bionic	986553169	Calle de los Abedules, 50	92465846	72.9
2	2	Forgeon	916349630	Calle Arboleda, 52	267509373	35.96
3	3	MSI	976842772	Avenida Roble, 30	833692133	0.0
4	4	Tempest	956332624	Calle Secuoya, 13	932561783	53.06
5	5	Xiaomi	936457594	Avenida Roble, 68	163489231	107.95

MSI                    976842772                    Avenida Roble, 30                    833692133                    0.0 €                    [Editar](#)

- o Edición de la información del producto por el administrador

La visualización es la misma que cuando accedemos desde proveedor.html:



Suministros Informaticos Home Edición productos del proveedor

**Edición del producto:**

Producto: MSI Max Teclado Gaming Mecánico Custom Media Dock RGB Modular Cherry MX Red

Tipo: Teclado

Cantidad: 4

Cantidad Requerida: 4

Precio: 79.95

IVA: 10.0

Descuento: 0.0

Dimensiones del producto: 30x20x2

Lugar en el que se encuentra el producto: Bodega

**Guardar cambios**

En el código de la función, producto\_admin(), y en el del editar\_producto\_admin.html cambian principalmente 2 cosas con respecto a las funciones y html que teníamos para los proveedores. Los mayores cambios son que al guardar los cambios nos devuelve a la página admin.html y que le debemos de enviar la información de todos los proveedores para que nos pueda mostrar todos. El código modificado, también con los métodos GET y POST para diferenciar sus funcionalidades, de ambos es el siguiente:



```
# Ruta y función para mostrar la información del producto seleccionado para editar por el administrador
@app.route("/info_productos_admin/<int:id>", methods=["GET"])
def producto_admin(id):
    producto_a_mostrar = info_producto(id)
    return render_template("editar_productos_admin.html", producto=producto_a_mostrar)

# Ruta y función para actualizar la información del producto editado por el administrador
@app.route("/info_productos_admin", methods=["POST"])
def producto_actualizado_admin():
    # Establecemos los campos de la tabla a poder ser modificados en este apartado
    id = request.form.get("id")
    nombre = request.form.get("nombre")
    tipo = request.form.get("tipo")
    cantidad = request.form.get("cantidad")
    cantidad_requerida = request.form.get("cantidad_requerida")
    precio = request.form.get("precio")
    descuento = request.form.get("descuento")
    dimension = request.form.get("dimension")
    lugar = request.form.get("lugar")
    # Filtramos la tabla de proveedores por el id
    datos_producto = db.session.query(Registro_productos).filter(Registro_productos.id == id)
    datos_producto.update(dict(nombre=nombre,
                               tipo=tipo,
                               cantidad=cantidad,
                               cantidad_requerida=cantidad_requerida,
                               precio=precio,
                               iva=iva,
                               descuento=descuento,
                               dimension=dimension,
                               lugar=lugar)) # Modificamos la información del proveedor en la base de datos
    db.session.commit() # Guardamos los cambios realizados
    # Leemos nuevamente la información modificada
    datos_proveedor = db.session.query(Registro_proveedores).all()
    todos_los_productos = db.session.query(Registro_productos).all()
    return render_template("admin.html", info_proveedor=datos_proveedor, lista_productos=todos_los_productos)
```

El código HTML es el mismo en editar\_productos.html que en editar\_productos\_admin.

El único cambio se encuentra en el action del form puesto que hemos puesto la ruta de "/info\_productos\_admin" para que al darle a guardar los cambios nos redirija a admin.html:

```
<!--Lista de los datos del proveedor-->
<h4>Edición del producto:</h4><br>
<!--Formulario para modificar la información del proveedor-->
<form action="/info_productos_admin" method="post">
    <input value="{{producto.id}}" type="hidden" class="form-control" id="id" name="id">
    <div class="mb-3">
```

- Muestra de que al guardar los cambios estos se modifican:

Base de datos antes de los cambios:



DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda\database\registros.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios Abrir proyecto Guardar proyecto

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: productos Filtrar en cualquier colu...

	nombre	tipo	cantidad	cantidad_requerida	precio	iva	de
1	Aerocool Bionic V2 RGB Cristal ...	Torre de ordenador	8	10	40.5	10.0	
2	Forgeon Acrux Leather Silla Gaming ...	Silla Gaming	7	8	149.95	10.0	
3	Forgeon Bolt PSU 750W 80+ Gold Ful...	Fuente de alimentación	12	12	53.2	10.0	
4	Forgeon Clutch Teclado Gaming RGB ...	Teclado	5	6	39.95	10.0	
5	MSI Max Teclado Gaming Mecánico ...	Teclado	4	4	79.95	10.0	
6	MSI Optix MAG245R2 23.8 pulgadas ...	Pantalla	4	4	160.0	10.0	
7	Bionic Sagitta RGB 650W 80 Plus Gol...	Fuente de alimentación	8	8	44.95	10.0	
8	Tempest X5S Strider RGB Ratón ...	Ratón	5	6	58.95	10.0	
9	Tempest X8 Keeper RGB Ratón Gami...	Ratón	4	4	72.45	10.0	
10	Xiaomi Mi Desktop Monitor 27 pulgad...	Pantalla	4	5	119.95	10.0	

Base de datos tras cambiar el precio y la cantidad requerida, con imagen de la modificación en admin.html:

DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almazán\_Lalmolda\database\registros.db

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios Abrir proyecto Guardar proyecto

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: productos Filtrar en cualquier colu...

	nombre	tipo	cantidad	cantidad_requerida	precio	iva	de
1	Aerocool Bionic V2 RGB Cristal ...	Torre de ordenador	8	10	40.5	10.0	
2	Forgeon Acrux Leather Silla Gaming ...	Silla Gaming	7	8	149.95	10.0	
3	Forgeon Bolt PSU 750W 80+ Gold Ful...	Fuente de alimentación	12	12	53.2	10.0	
4	Forgeon Clutch Teclado Gaming RGB ...	Teclado	5	6	39.95	10.0	
5	MSI Max Teclado Gaming Mecánico ...	Teclado	4	6	69.95	10.0	
6	MSI Optix MAG245R2 23.8 pulgadas ...	Pantalla	4	4	160.0	10.0	
7	Bionic Sagitta RGB 650W 80 Plus Gol...	Fuente de alimentación	8	8	44.95	10.0	
8	Tempest X5S Strider RGB Ratón ...	Ratón	5	6	58.95	10.0	
9	Tempest X8 Keeper RGB Ratón Gami...	Ratón	4	4	72.45	10.0	
10	Xiaomi Mi Desktop Monitor 27 pulgad...	Pantalla	4	5	119.95	10.0	



MSI	MSI Max Teclado Gaming Mecánico Custom Media Dock RGB Modular Cherry MX Red	Teclado	4%	6	69.95	10.0	0.0	30x20x2	Bodega	<button>Editar</button>
-----	--	---------	----	---	-------	------	-----	---------	--------	-------------------------

11. Función, en nuestro main.py, para añadir en la tabla proveedores la columna facturación, con el valor total de lo que ha ganado cada uno de los proveedores por la venta de cada uno de sus productos. Este valor llegará a facturación habiéndole restado el iva del producto y el descuento que le haya querido poner el proveedor, en caso de haber puesto uno:

```
# Función para sumar el valor de cada producto comprado para que se sepa la cantidad que factura el proveedor
def facturacion_proveedor():
    # Agrupamos en una suma el valor de lo que se ha vendido por cada producto
    valor_facturacion = db.session.query(Facturacion_detalle.id_producto, func.sum(Facturacion_detalle.valor)).group_by(Facturacion_detalle.id_producto).all()
    for facturacion in valor_facturacion:# Iteramos para obtener la información total de cada producto
        id_producto = facturacion[0]# Obtenemos el id del producto
        valor = facturacion[1].# Obtenemos la suma de los valores vendidos por el producto
        datos_producto = info_producto(id_producto) # Obtenemos la información del producto para saber quién es el proveedor
        # Actualizamos el campo facturacion para que cada proveedor sepa la cantidad que obtuvo de ganancias, tras restar iva y descuentos
        db.session.query(Registro_proveedores).filter(Registro_proveedores.id==datos_producto.id_proveedor).update(dict(facturacion=valor))
    db.session.commit() # Guardamos los cambios
```

La añadimos a nuestro home(), en el main.py, para que nos la inicialice y podamos ver la cantidad que cada proveedor facturó. Imagen de nuestra base de datos, tabla de proveedores, para que se vea ya se encuentran de todas las compras realizadas los valores que facturó cada proveedor por ellas:

DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_Final\_Santiago\_Almañan\_Lalmolda\dat

Archivo Editar Ver Herramientas Ayuda

Nueva base de datos Abrir base de datos Guardar cambios Deshacer cambios Abrir |

Estructura Hoja de datos Editar pragmas Ejecutar SQL

Tabla: proveedores Filtrar en cualquier

	id	proveedor	telefono	direccion	cif	facturacion
	Filtro	Filtro	Filtro	Filtro	Filtro	Filtro
1	1	Bionic	986553169	Calle de los Abedules, 50	92465846	72.9
2	2	Forgeon	916349630	Calle Arboleda, 52	267509373	35.96
3	3	MSI	976842772	Avenida Roble, 30	833692133	0.0
4	4	Tempest	956332624	Calle Secuoya, 13	932561783	53.06
5	5	Xiaomi	936457594	Avenida Roble, 68	163489231	107.95



Imagen de la tabla de facturación\_detalle para poder de donde han salido los valores de la facturación de cada proveedor:

The screenshot shows the DB Browser for SQLite interface with the title bar "DB Browser for SQLite - C:\Users\siris\PycharmProjects\Proyecto\_". The menu bar includes Archivo, Editar, Ver, Herramientas, and Ayuda. Below the menu is a toolbar with icons for Nueva base de datos, Abrir base de datos, and Guardar cambios. A tab bar at the top has Estructura, Hoja de datos, Editar pragmas, and Ejecutar SQL, with "facturacion\_detalle" selected. Below the tabs is a toolbar with icons for search, sort, and export. The main area displays a table with the following data:

	id	id_factura	id_producto	cantidad	valor	
	Filtro	Filtro	Filtro	Filtro	Filtro	
1	7	4	1	1	36.45	
2	8	4	2	1	121.45	
3	9	4	4	1	35.96	
4	10	5	1	1	36.45	
5	11	5	8	1	53.06	
6	12	5	10	1	107.95	

12. Configuración de las tablas para que a los proveedores y a los compradores se les muestre la cantidad de productos que han sido vendidos:

Realizamos la importación de las librerías que vamos a necesitar para realizar las tablas:

```
import matplotlib.pyplot as plt
from io import BytesIO
import base64
```

Código de cada una de las tablas en el main.py. Tenemos que realizar la conversión a imagen de cada una de ellas ya que html no puede imprimirlas de forma directa tras configurarlas con Matplotlib:



```
# Función para crear la gráfica que muestre a los compradores los productos que son vendidos
def grafica_ventas():
    fig, ax = plt.subplots()

    # Obtenemos un listado de los productos vendidos
    productos_vendidos = db.session.query(Facturacion_detalle.id_producto, func.count(Facturacion_detalle.id_producto)).group_by(Facturacion_detalle.id_producto).all()
    lista_id_productos_vendidos = []
    lista_cantidad = []
    for productos in productos_vendidos:
        lista_id_productos_vendidos.append(productos[0])
        lista_cantidad.append(productos[1])

    ax.bar(lista_id_productos_vendidos, lista_cantidad)

    ax.set_ylabel('Cantidad vendida por producto')
    ax.set_xlabel('Cantidad de productos vendidos')
    ax.set_title('Productos vendidos')

    # Convertimos la gráfica en una imagen
    img = BytesIO() # Creamos una imagen vacía
    plt.savefig(img, format='png') # Guardamos la gráfica en la imagen previa
    plt.close() # Cerramos Matplotlib
    img.seek(0) # Cerramos la imagen
    plot_url = base64.b64encode(img.getvalue()).decode('utf8') # Codificamos la imagen para poder mostrarla en el html

    return plot_url # Retornamos la gráfica codificada

# Función que muestra a los proveedores cuáles de sus productos son los más comprados
def grafica_compras():
    fig, ax = plt.subplots()

    # Obtenemos un listado de los productos vendidos
    productos_vendidos = db.session.query(Facturacion_detalle.id_producto, func.count(Facturacion_detalle.id_producto)).group_by(Facturacion_detalle.id_producto).all()
    lista_id_productos_vendidos = []
    lista_cantidad = []
    for productos in productos_vendidos:
        lista_id_productos_vendidos.append(productos[0])
        lista_cantidad.append(productos[1])

    ax.bar(lista_id_productos_vendidos, lista_cantidad)

    ax.set_ylabel('Cantidad vendida por producto')
    ax.set_title('Productos vendidos')
    ax.legend(title='Cantidad de productos vendidos')

    # Convertimos la gráfica en una imagen
    img = BytesIO() # Creamos una imagen vacía
    plt.savefig(img, format='png') # Guardamos la gráfica en la imagen previa
    plt.close() # Cerramos Matplotlib
    img.seek(0) # Cerramos la imagen
    plot_url = base64.b64encode(img.getvalue()).decode('utf8') # Codificamos la imagen para poder mostrarla en el html

    return plot_url # Retornamos la gráfica codificada
```

Código, en nuestra ruta home(), e imagen que introducimos en nuestro index.html para que la gráfica se muestre a los compradores y puedan saber la cantidad vendida por cada producto. En el eje x decidí poner los números de id de los productos ya que los nombres son muy



largos y no puedo introducirlos en la gráfica. Por ello pongo un listado de referencia antes de la gráfica para que puedan ser identificados:

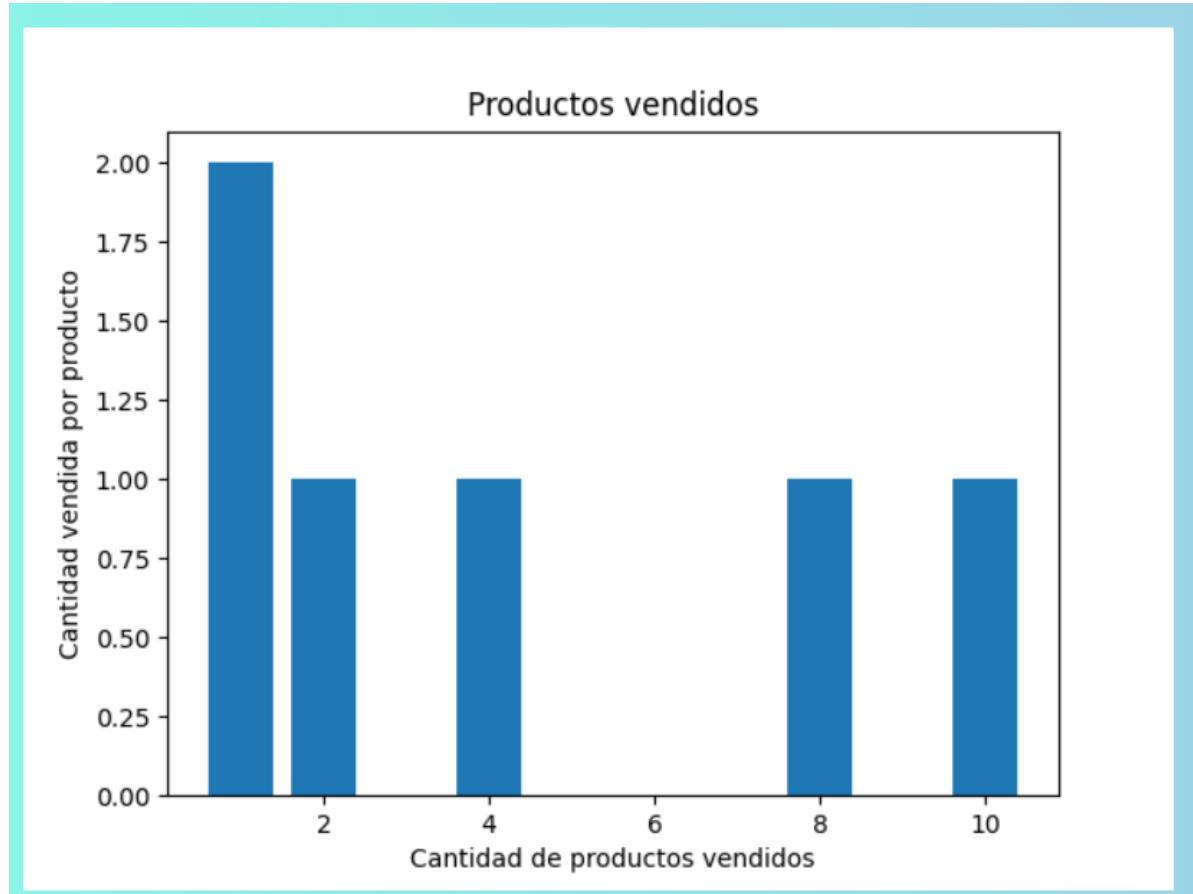
```
# Creamos a la función 'home'  
@app.route("/")  
def home():  
    # Variable "lista_de_productos" para que se nos de la información de todos los productos  
    lista_de_productos = todos_los_productos()  
    facturacion_proveedor()  
    grafica = grafica_ventas()  
    return render_template("index.html", productos=lista_de_productos, grafica=grafica)
```

```
<h4>Productos vendidos:</h4><br>  
<!--Mostramos la gráfica de ventas--&gt;<br/><ul>  
    {%for producto in productos%}  
        <li>{{producto.id}} => {{producto.nombre}}</li>  
    {%endfor%}  
    </ul>  
      
    .
```

Imagen del resultado en nuestra app web:

## Productos vendidos:

- 1 => Aerocool Bionic V2 RGB Cristal Templado USB 3.0 Negra
- 2 => Forgeon Acrux Leather Silla Gaming Negra
- 3 => Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación
- 4 => Forgeon Clutch Teclado Gaming RGB 60% Switch Blue
- 5 => MSI Max Teclado Gaming Mecánico Custom Media Dock RGB Modular Cherry MX Red
- 6 => MSI Optix MAG245R2 23.8 pulgadas LED IPS FullHD 170Hz FreeSync Premium
- 7 => Bionic Sagitta RGB 650W 80 Plus Gold Full Modular
- 8 => Tempest X5S Strider RGB Ratón Gaming 4.800 DPI Negro
- 9 => Tempest X8 Keeper RGB Ratón Gaming 10.000 DPI Negro
- 10 => Xiaomi Mi Desktop Monitor 27 pulgadas LED IPS FullHD 75Hz



En la muestra de la tabla para proveedores las diferencias son que en la lista previa a la tabal vamos a iterar con el id del proveedor para que le muestre una lista de los productos que tiene a la venta y pueda identificarlos en la tabla. Mostramos el código en proveedores.html:

```

<h4>Información de Ventas:</h4><br>
<!-- Mostramos la gráfica de ventas--&gt;
&lt;ul&gt;
    {%for producto in lista_productos%}
        &lt;li&gt;{{producto.id}} =&gt; {{producto.nombre}}&lt;/li&gt;
    {%endfor%}
&lt;/ul&gt;
&lt;img src="data:image/png;base64, {{ grafica }}"&gt;
&lt;/section&gt;</pre>

```

Añadimos a nuestras rutas, en el main.py, la gráfica para que al acceder o al retornar de cada uno de los puntos nos la siga mostrando en proveedores.html:

- En la función login\_proveedores(), para que al acceder se nos muestre:



```
grafica = grafica_compras() # mostramos la gráfica
return render_template("proveedores.html", proveedor=proveedor, info_proveedor=datos_proveedor, lista_productos=todos_los_productos, grafica=grafica)
```

- En la función proveedor\_actualizado(), para que al volver de editar la información del proveedor se nos muestre:

```
grafica = grafica_compras() # mostramos la gráfica
return render_template("proveedores.html", info_proveedor=proveedor, lista_productos=todos_los_productos, grafica=grafica)
```

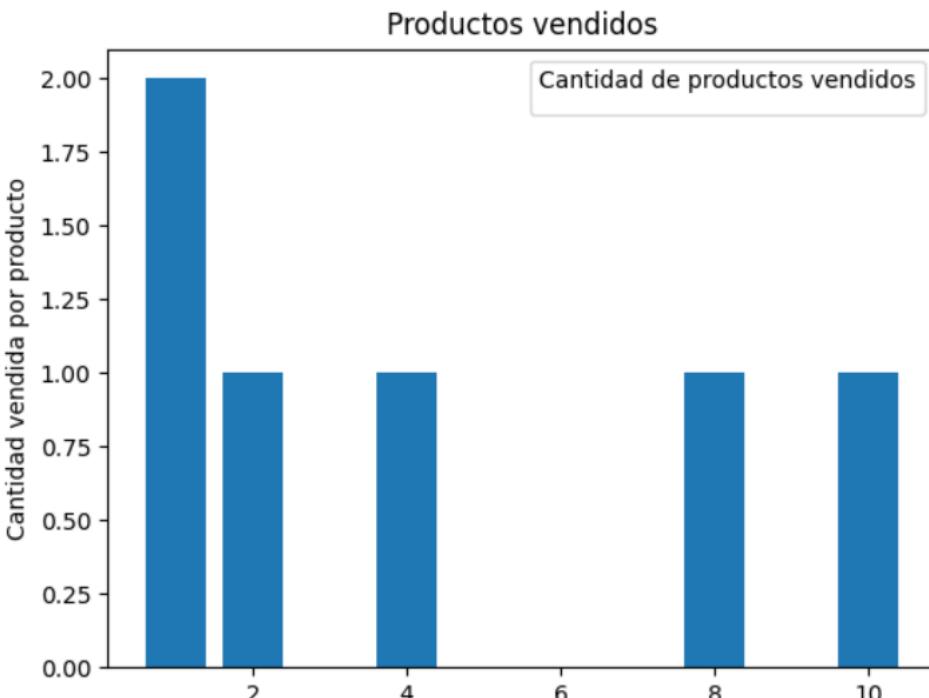
- En la función producto\_actualizado(), para que al volver de editar la información del producto se nos muestre:

```
grafica = grafica_compras() # mostramos la gráfica
return render_template("proveedores.html", info_proveedor=proveedor, lista_productos=todos_los_productos, grafica=grafica)
```

Imagen de como se ve la tabla en proveedores.html cuando nuestra app web esta en ejecución.

## Información de Ventas:

- 2 => Forgeon Acrux Leather Silla Gaming Negra
- 3 => Forgeon Bolt PSU 750W 80+ Gold Full Modular Fuente de Alimentación
- 4 => Forgeon Clutch Teclado Gaming RGB 60% Switch Blue





13. Finalmente, declaramos nuestra función principal e inicializamos la aplicación web desde nuestro main.py:

```
# Función principal
if __name__ == "__main__":
    db.Base.metadata.create_all(db.engine)
    app.run(debug=True) # El método 'app.run' nos inicia el servidor web
```

## CONCLUSIONES

La aplicación web consiste en:

- una página principal, donde mostramos los productos a la vista de los compradores para que puedan realizar sus compras,
- un carrito de compras, para que los compradores tengan temporalmente los artículos hasta que finalicen su proceso de compra o desechen el carrito y no compren,
- una barra de navegación que cohesiona toda la experiencia a lo largo de la navegación por la aplicación web,
- accesos, tanto para cada uno de los proveedores como para el administrador, con distintos niveles de alcance, pues el administrador debe de alcanzar a todo y cada proveedor solo alcanza a lo suyo,
- unas tablas que nos muestran la cantidad vendida de cada producto, brindando una información adicional tanto a compradores como a proveedores,
- y, finalmente, una base de datos, con distintas tablas, que nos sirve para poder mostrar e interconectar toda la información que requerimos.

Durante el proyecto hemos utilizado rutas, funciones, herencia y muchos recursos más con el fin de tener el producto final de esta aplicación web.