

---

# Rumbo al Proyecto final Full Stack Python con objetos y API REST

---

## Descripción general del proyecto:

El proyecto se trata de un sistema de inventario y carrito de compras implementado como una API en Python utilizando el framework Flask y una base de datos. El objetivo principal del proyecto es gestionar un inventario de productos y permitir a los usuarios agregar productos al carrito de compras.

Aquí hay un resumen de las principales características y funcionalidades del proyecto:

### Gestión de productos:

- Agregar un nuevo producto al inventario.
- Modificar la información de un producto existente en el inventario.
- Eliminar un producto del inventario.
- Consultar información de un producto por su código.

### Gestión del carrito de compras:

- Agregar productos al carrito de compras.
- Quitar productos del carrito de compras.
- Mostrar el contenido actual del carrito de compras.

### Persistencia de datos:

- Los datos de los productos se almacenan en una base de datos SQL.
- Se utiliza una conexión a la base de datos para realizar operaciones CRUD en los productos.
- El código proporcionado incluye las clases Producto, Inventario y Carrito, que representan la estructura y funcionalidad relacionada con los productos, el inventario y el carrito de compras, respectivamente. Además, se define una serie de rutas en Flask para manejar las solicitudes HTTP relacionadas con la gestión de productos y el carrito de compras.

Se implementan desde cero el backend y el frontend. En el caso del backend, el proyecto va "evolucionando", comenzando en el desarrollo de las funciones que se necesitan para manipular los productos y el carrito de compras utilizando arreglos en memoria, luego se modifica para utilizar objetos, más tarde se gestiona la persistencia de los datos utilizando una base de datos SQL, después se implementa la API en Flask y se aloja el script Python en un servidor, y por último se crea un frontend básico para interactuar con los datos desde el navegador, a través de la API creada.

El proyecto se divide en seis etapas:

**Primera etapa:** Implementar un CRUD de productos y un carrito de compras utilizando arreglos y funciones.

**Segunda etapa:** Convertir las funciones vistas en la clase anterior en objetos y clases.

**Tercera etapa:** Utilizar como almacenamiento una base de datos SQL.

**Cuarta etapa:** Implementar una API

**Quinta etapa:** Desplegar el proyecto en PythonAnywhere

**Sexta etapa:** Implementar un frontend


## TODO list

Está claro que esta API, y el proyecto en general, está lejos de ser una aplicación web completa. Este ejercicio puede ser mejorado, por ejemplo en los siguientes aspectos:

### 1) Utilizar una base de datos SQL en Pythonanywhere

Para ello, se debe acceder a la sección "databases" y configurar allí nuestra base de datos:

Send feedback Forums Help Blog Account Log out

 pythonanywhere  
by ANACONDA.

Dashboard Consoles Files Web Tasks **Databases**

MySQL

Postgres

## Initialize MySQL

Let's get started! The first thing to do is to initialize a MySQL server:

Enter a new password in the form below, and note it down: you'll need it to access the databases once you've created them. You will only need to do this once.

**New password:**

**Confirm password:**

**Initialize MySQL**

This should be different to your main PythonAnywhere password, because it is likely to appear in plain text in any web applications you write.

Y, por supuesto, el código Python desarrollado debe modificarse. Hay que utilizar el módulo **pymysql** en lugar de **sqlite3**, y modificar la forma en que se accede a la base de datos.

Se deja a disposición del alumno un ejemplo de código en **Clase 03\_pymysql.py**. Recuerda que para utilizar ese código, necesitas tener funcionando el servidor SQL (si lo vas a utilizar en forma local) o la base de datos correctamente configurada en Pythonanywhere.

**2) No es difícil incorporar mas campos al inventario, o mas tablas a nuestra base de datos.** El código desarrollado sirve de ejemplo de como implementar estas mejoras. Incluso se pueden agregar imágenes, implementando un mecanismo que permite seleccionar una imagen de la computadora del usuario y que las suba al servidor. En la base de datos se guarda el nombre del archivo unicamente.

En ese caso, debemos utilizar algún servidor que permita almacenar un mayor número de archivos (Pythonanywhere, en su versión gratuita, solo permite 5 archivos por sitio).

**3) El carrito de compras es el mismo para todos los usuarios.** Para que cada usuario tenga su propio carrito de compras es necesario, por supuesto, definir la clase *Cliente* (o similar) y crear una tabla en la que se almace el contenido de los carritos de compras de los usuarios con un campo que sirva de clave externa para vincular cada uno de sus registros con el usuario correspondiente.