

El Gran Sistema de Gestión de Inventario

¡Hola desarrolladores!

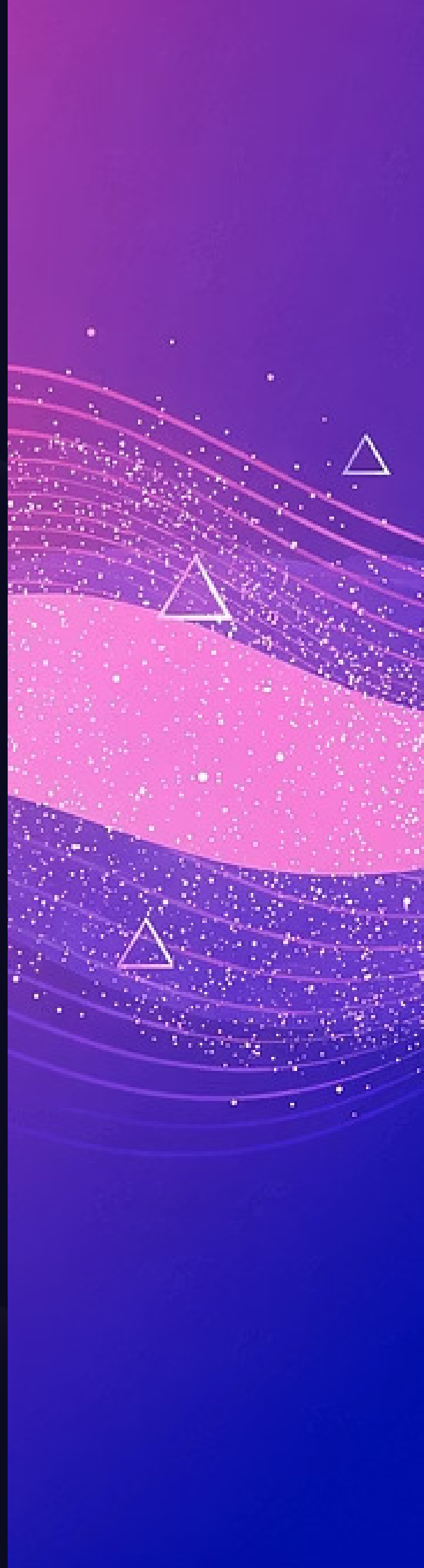
Han llegado al momento de la verdad. Ya saben manipular datos con SQL (DML) y entienden la importancia de viajar en el tiempo si algo sale mal (Transacciones). Ahora, vamos a unir el mundo de **Python** con el mundo de **MySQL**.

Vamos a construir una aplicación real: un **Sistema de Gestión de Inventario**. Imaginen que son los encargados de tecnología de una tienda gigante y necesitan una herramienta para agregar productos, actualizar precios, eliminar ítems obsoletos y consultar el stock.

¡No se asusten! Lo haremos pieza por pieza.

Bootcamp Desarrollo de Aplicaciones Full Stack Python

*Instructor **Felipe Cuevas***



Requisitos Previos

Antes de escribir una sola línea de código, asegúrate de tener tu entorno listo:

Docker

Corriendo con tu contenedor de MySQL.

Python

Instalado en tu computadora.

Librería MySQL

La librería mysql-connector-python instalada (pip install mysql-connector-python).

Cliente SQL

Tu cliente SQL listo.

1. Aquellos que usan PyCharm solo necesitan configurar la conexión BD.

2. Aquellos que usan Visual Studio Code pueden usar el plugin **SQLTools**.

Bootcamp Desarrollo de Aplicaciones Full Stack Python

Instructor [Felipe Cuevas](#)

PASO 1: Preparar el Terreno (La Base de Datos)

Lo primero es crear donde vivirán nuestros datos. He preparado un script mágico para ustedes.

01	02	03
Descarga el archivo	Ejecuta el script	¡Magia instantánea!
Descarga el archivo script.txt adjunto a esta actividad.	Abre su contenido y ejecútalo en tu gestor de base de datos conectado a Docker.	¿Qué pasará? Se creará la base de datos Ejercicio_BD y la tabla Productos llena con 200 registros de prueba. ¡Magia instantánea!

Bootcamp Desarrollo de Aplicaciones Full Stack Python

Instructor **Felipe Cuevas**

PASO 2: La Clase para Producto

En Python, necesitamos un "molde" para manejar la información que viene de la base de datos.

- 📌 **Tu Misión:** Crea un archivo llamado `Producto.py`. Dentro, crea la clase `Producto` que represente la tabla que acabas de crear.

Debe cumplir con:

Constructor (`__init__`)

Debe recibir id (puede ser `None` si es nuevo), nombre, categoría, marca, precio, stock, descuento, peso, fecha_ingreso.

Getters y Setters

Métodos para obtener y modificar cada atributo (ej. `get_nombre()`, `set_precio()`).
Nota: Aunque en Python se usan decoradores, hazlo con métodos explícitos para practicar la lógica.

Método `str`

Para que al imprimir el objeto, se vea bonito y legible (ej: `"Producto: Laptop HP - Precio: $1599.99"`).

Bootcamp Desarrollo de Aplicaciones Full Stack Python

Instructor [Felipe Cuevas](#)

PASO 3: El Motor con Seguridad ACID

Aquí es donde aplicamos lo aprendido sobre **Transacciones**. Crearemos un archivo llamado BaseDatos.py. Este será el único archivo con permiso para hablar con MySQL.

Tu Misión: Crear funciones para conectar y realizar el CRUD.

Requisitos OBLIGATORIOS de Seguridad (Transacciones)

Para las funciones de **Registrar**, **Actualizar** y **Eliminar**, debes usar la estructura de seguridad:

1. **try:** Inicias la transacción. Intentas ejecutar la orden SQL.
2. **connection.commit():** Si todo sale bien, confirmas los cambios (¡Guardado en piedra!).
3. **except:** Si algo falla, atrapas el error e invocas **connection.rollback()**. (¡Deshacer el tiempo!).
4. **finally:** Cierras el cursor y la conexión.

Las funciones que debes crear:

- `conectar_bd():` Retorna la conexión.
- `listar_productos():` Retorna una lista de objetos Producto.
- `registrar_producto(producto):` Recibe un objeto Producto y lo guarda (INSERT).
- `actualizar_producto(producto):` Recibe un objeto con los nuevos datos y actualiza en la BD (UPDATE).
¡No olvides el WHERE por ID!
- `eliminar_producto(id_producto):` Elimina el producto por su ID (DELETE).

Bootcamp Desarrollo de Aplicaciones Full Stack Python

Instructor [Felipe Cuevas](#)

PASO 4: El Control de Mando

Finalmente, necesitamos una forma de usar esto. Crearemos el archivo Menu.py.

Requisitos Especiales:

1

Diccionario de Funciones

No quiero una escalera infinita de if/else. Debes usar un **diccionario** para manejar las opciones.

- *Ejemplo:* opciones =
 { '1': funcion_listar,
 '2': funcion_crear ... }

2

Menú Cíclico

La app no debe cerrarse hasta que el usuario elija "Salir".

3

Interacción

Aquí pedirás los datos al usuario (input), crearás el objeto Producto y se lo pasarás a las funciones de tu archivo BaseDatos.py.

Estructura sugerida del Menú:

```
=== GESTIÓN DE INVENTARIO ===  
1. Listar productos  
2. Registrar nuevo producto  
3. Actualizar producto existente  
4. Eliminar producto  
5. Salir
```

Bootcamp Desarrollo de Aplicaciones Full Stack Python

Instructor [Felipe Cuevas](#)

Ejemplo de Estructura de Transacción (Ayuda del Mentor)

Sé que el manejo de transacciones puede intimidar al principio. Aquí tienes un "esqueleto" de cómo debería verse tu función de insertar en BaseDatos.py para cumplir con ACID:

```
def registrar_producto(producto):
    conexion = conectar_bd()
    if conexion:
        try:
            cursor = conexion.cursor()
            # OJO: Desactivamos autocommit implícitamente al usar
            # commit/rollback manual
            sql = "INSERT INTO Productos (Nombre, Categoria, ...) VALUES (%s, %s, ...)"
            valores = (producto.get_nombre(), producto.get_categoria(), ...)
            cursor.execute(sql, valores)
            conexion.commit() # <--- ¡AQUÍ SE CONFIRMA! (Durabilidad)
            print("¡Producto guardado exitosamente!")
        except Exception as e:
            conexion.rollback() # <--- ¡AQUÍ SE DESHACE SI HAY ERROR!
            print(f"Error al guardar: {e}")
        finally:
            cursor.close()
            conexion.close()
```

Instrucciones de Entrega

¡Hora de mostrar tu trabajo al mundo!



Git & GitHub

Inicializa un repositorio en tu carpeta del proyecto.



Commit

Haz commits con mensajes claros (ej: "Agregando clase Producto", "Implementando CRUD con rollback").



Push

Sube tu proyecto a un repositorio público en GitHub.



El Correo

Envía tu entrega.

Detalles del correo:

- **Destinatario:** felipe@ecas-otec.cl
- **Asunto:** Entrega Proyecto Final Inventario - [TU NOMBRE]
- **Contenido:** Un saludo cordial y el **link directo a tu repositorio**.



Consejo Final: No intentes hacer todo de una vez. Primero logra que la conexión funcione. Luego intenta listar los productos. Después intenta insertar uno. ¡Divide y vencerás!