«Talento Tech»

Iniciación a la Programación con Python

CLASE 15

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 15 | SQLite

Temario:

- Definimos y creamos la base de datos
- Agregamos a las funciones creadas antes de las consultas de SQL
- Revisión de los criterios de evaluación.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Introducción.

Llegamos a una clase fundamental en el recorrido de este curso: la última antes de la entrega del Trabajo Final Integrador (TFI). Vamos a concentrarnos en brindarte las herramientas y el acompañamiento necesarios para que puedas encarar y completar tu proyecto de forma independiente. Esta clase está diseñada para orientarte y ayudarte a consolidar todo lo que venimos aprendiendo.

Para hacer las explicaciones más claras y prácticas, vamos a trabajar con ejemplos en una base de datos, llamada Personas, diferente a la que seguramente implementarás en tu TFI, con el objetivo de que sea más sencilla. La estructura de esta base de datos, que ya vimos en la clase anterior, contiene información sobre personas, incluyendo su nombre, edad y ciudad. Usar esta tabla nos permitirá ilustrar conceptos y operaciones similares a las que necesitarás implementar en el TFI, pero sin resolver directamente el trabajo que te corresponde.

Cada función que desarrollaremos y cada operación que veremos pueden ser utilizadas como modelo para que armes tus propias funciones en el TFI. Este enfoque te permitirá practicar y entender a fondo cómo trabajar con bases de datos en Python, sin copiar literalmente el código que necesitas en tu proyecto final, sino adaptándolo y haciéndolo tuyo. Así, la clase de hoy será una especie de "guía práctica" que podrás consultar para avanzar con seguridad y autonomía en tu TFI.

Registrar personas.

Esta función permitirá agregar una nueva persona a la tabla Personas en nuestra base de datos. Al igual que en el TFI deberán hacerlo para registrar productos, el proceso aquí consiste en pedir los datos necesarios para el nuevo registro e insertarlos en la base de datos. La estructura de la tabla que trabajaremos contiene los campos nombre, edad y ciudad, todos relevantes para ver cómo manejar entradas de texto y números enteros.

Primero, pedimos los datos al usuario o usuaria, incluyendo el nombre, la edad y la ciudad de residencia de la persona. Luego, usamos la instrucción INSERT para agregar esos datos a la tabla.

```
def registrar_persona():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Pedir los datos de la persona
    nombre = input("Ingrese el nombre de la persona: ")
    edad = int(input("Ingrese la edad de la persona: "))
    ciudad = input("Ingrese la ciudad de residencia: ")

    # Insertar los datos en la tabla Personas
    cursor.execute("INSERT INTO Personas (nombre, edad, ciudad) VALUES
(?, ?, ?)", (nombre, edad, ciudad))

    # Confirmar la inserción
    conexion.commit()
    print("Persona registrada con éxito.")

    # Cerrar la conexión
    conexion.close()
```

Este código realiza la conexión a la base de datos y crea un cursor, que se utiliza para ejecutar la consulta SQL INSERT INTO Personas (nombre, edad, ciudad) VALUES (?, ?, ?). Los signos de interrogación (“?”) son marcadores de posición para cada valor que se insertará en los campos correspondientes. Después de ejecutar la consulta, commit() guarda los cambios en la base de datos y, finalmente, cerramos la conexión con close().

Este ejemplo de registrar_persona() puede adaptarse para el TFI al cambiar los campos y la estructura de la tabla, siguiendo la misma lógica. Una vez que tengamos implementadas todas las funciones necesarias, crearemos un menú que permita invocar cada una de ellas de forma sencilla. Al estructurar el programa de esta manera, facilitaremos la navegación y el acceso a cada función sin tener que ejecutarlas manualmente.

Mostrar datos de las personas.

La función mostrar_personas() nos permitirá visualizar todos los registros de la tabla Personas en la base de datos. Esto es similar a la funcionalidad de mostrar productos en el TFI, pero aquí, en lugar de productos, estaremos viendo los datos de las personas almacenadas. Esta función será muy útil para que se puedan revisar fácilmente todos los registros de una vez, algo especialmente importante en la gestión de un inventario o de cualquier base de datos de uso frecuente.

Para implementar esta función, realizamos una consulta SELECT * FROM Personas, que recupera todos los registros de la tabla. Luego, mostramos cada registro en pantalla, de la siguiente manera:

```
def mostrar_personas():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Recuperar todos los registros de la tabla Personas
    cursor.execute("SELECT * FROM Personas")
    resultados = cursor.fetchall()

    # Verificar si la tabla tiene registros
    if resultados:
        print("Listado de personas registradas:")
        for registro in resultados:
```

```
            print("Nombre:", registro[0], "Edad:", registro[1],
"Ciudad:", registro[2])
    else:
        print("No hay registros en la tabla Personas.")

    # Cerrar la conexión
    conexion.close()
```

En esta función, primero establecemos la conexión a la base de datos y creamos el cursor. La consulta SELECT * FROM Personas selecciona todos los registros de la tabla. Luego, con fetchall(), obtenemos una lista de todos los resultados y, si la tabla tiene registros, los mostramos recorriendo cada uno con un bucle for. Finalmente, cerramos la conexión.

Este mismo enfoque es adaptable al TFI: en lugar de mostrar personas, podrás mostrar productos del inventario, cambiando los nombres de los campos y la tabla según tu proyecto.

Actualizar datos.

La función actualizar_persona() permite modificar la información de una persona específica en la base de datos. En este caso, nos enfocaremos en actualizar la edad de una persona. Este proceso es similar a lo que tendrás que hacer en el TFI cuando necesites actualizar información de un producto. Aquí, pediremos el nombre de la persona cuyo registro queremos modificar y la nueva edad que deseamos guardar.

Para implementar esta función, realizamos una consulta UPDATE que afecta solo al registro especificado por la usuaria o usuario. La función se asegura de que la persona exista antes de aplicar el cambio.

```
def actualizar_persona():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Pedir el nombre de la persona y la nueva edad
    nombre = input("Ingrese el nombre de la persona que desea
actualizar: ")
    nueva_edad = int(input("Ingrese la nueva edad: "))

    # Ejecutar la consulta de actualización
    cursor.execute("UPDATE Personas SET edad = ? WHERE nombre = ?",
(nueva_edad, nombre))

    # Verificar si se realizó algún cambio
    if cursor.rowcount > 0:
        print("La edad de", nombre, "ha sido actualizada con éxito.")
    else:
        print("No se encontró a una persona con ese nombre.")

    # Guardar los cambios
    conexion.commit()

    # Cerrar la conexión
    conexion.close()
```

Aquí, la función establece la conexión a la base de datos y luego solicita al usuario que ingrese el nombre de la persona y la nueva edad. La consulta **UPDATE Personas SET edad = ? WHERE nombre = ?** asegura que sólo se actualice el campo edad de la persona indicada. Después de ejecutar la consulta, **cursor.rowcount** verifica si se realizó un

cambio: si es mayor a cero, significa que la actualización fue exitosa. Finalmente, los cambios se guardan y la conexión se cierra.

Borrar registros.

La función **eliminar_persona()** permite borrar un registro específico de la tabla Personas en la base de datos. En este caso, te pedirá que indiques el nombre de la persona que deseás eliminar y, si se encuentra un registro con ese nombre, se eliminará de forma definitiva. Este procedimiento es similar al que realizarás en el TFI cuando necesites eliminar productos del inventario.

Para implementar esta función, realizamos una consulta DELETE que elimina únicamente el registro especificado. Además, verificamos si la eliminación fue exitosa.

```
def eliminar_persona():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Pedir el nombre de la persona a eliminar
    nombre = input("Ingrese el nombre de la persona que desea eliminar:
")

    # Ejecutar la consulta de eliminación
    cursor.execute("DELETE FROM Personas WHERE nombre = ?", (nombre,))

    # Verificar si se eliminó algún registro
    if cursor.rowcount > 0:
        print("La persona", nombre, "ha sido eliminada con éxito.")
    else:
        print("No se encontró a una persona con ese nombre.")
```

```
# Guardar los cambios
conexion.commit()

# Cerrar la conexión
conexion.close()
```

En esta función, primero conectamos a la base de datos y solicitamos el nombre de la persona que queremos eliminar. La consulta **DELETE FROM Personas WHERE nombre = ?** se ejecuta sólo para el registro que coincida con el nombre ingresado. Después, usamos **cursor.rowcount** para verificar si se eliminó algún registro: si es mayor a cero, confirmamos que la eliminación fue exitosa. Finalmente, guardamos los cambios y cerramos la conexión.

Para el TFI, sólo necesitarás ajustar la estructura de la tabla y los nombres de los campos, aplicando la misma lógica para asegurarte de que el registro específico se elimine correctamente.

Buscar datos.

La función **buscar_persona()** permite consultar un registro específico, recuperando los datos de una persona en función de su nombre. Esta funcionalidad es útil para localizar información detallada de un registro particular sin necesidad de ver todos los datos. En el TFI, necesitarás una función similar para buscar productos, adaptando esta estructura a tu proyecto.

Para implementar esta función, utilizamos una consulta SELECT con una condición WHERE para filtrar sólo el registro que corresponde al nombre ingresado por el usuario o usuaria.

```python
def buscar_persona():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Pedir el nombre de la persona que se desea buscar
    nombre = input("Ingrese el nombre de la persona que desea buscar: ")

    # Ejecutar la consulta de búsqueda
    cursor.execute("SELECT * FROM Personas WHERE nombre = ?", (nombre,))
    resultado = cursor.fetchone()

    # Verificar si se encontró el registro
    if resultado:
        print("Información de la persona encontrada:")
        print("Nombre:", resultado[0])
        print("Edad:", resultado[1])
        print("Ciudad:", resultado[2])
    else:
        print("No se encontró a una persona con ese nombre.")

    # Cerrar la conexión
    conexion.close()
```

En esta función, primero establecemos la conexión a la base de datos y solicitamos el nombre de la persona que deseamos buscar. Luego ejecutamos **SELECT * FROM Personas WHERE nombre = ?**, y con **fetchone()** obtenemos el primer resultado que coincida con el nombre ingresado. Si el registro existe, lo mostramos en pantalla, mostrando cada campo de forma clara. En caso contrario, indicamos que no se encontró el registro. Por último, cerramos la conexión.

Esta misma lógica puede ser útil para buscar productos en el inventario del TFI.

Listado de personas menores de edad.

La función **reporte_menores_edad()** generará un reporte de todas las personas en la base de datos que son menores de edad, es decir, aquellas cuya edad es menor a 18 años. Esta función es útil para practicar cómo aplicar condiciones en una consulta SQL y mostrar únicamente los registros que cumplen con un criterio específico. En el TFI, podrás adaptar esta lógica para crear reportes según el criterio que necesites, como productos en bajo stock.

En esta función, usamos la consulta SELECT con una cláusula WHERE que filtra sólo los registros de personas con edad inferior a 18.

```
def reporte_menores_edad():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Ejecutar la consulta para obtener personas menores de 18 años
    cursor.execute("SELECT * FROM Personas WHERE edad < 18")
    resultados = cursor.fetchall()

    # Verificar si hay resultados y mostrarlos
    if resultados:
        print("Personas menores de edad:")
        for registro in resultados:
            print("Nombre:", registro[0], "Edad:", registro[1],
"Ciudad:", registro[2])
    else:
        print("No se encontraron personas menores de edad.")
```

```
    # Cerrar la conexión
    conexion.close()
```

En esta función, primero establecemos la conexión a la base de datos y luego ejecutamos la consulta **SELECT * FROM Personas WHERE edad < 18,** que selecciona a todas las personas con edad menor a 18 años. Usamos **fetchall()** para obtener todos los registros que cumplen con esta condición. Si hay resultados, mostramos cada persona con su nombre, edad y ciudad; de lo contrario, indicamos que no se encontraron personas menores de edad. Por último, cerramos la conexión.

Este reporte de personas menores de edad puede adaptarse en el TFI para identificar productos en condiciones específicas, como en bajo stock.

Menú principal.

Una vez que hemos implementado todas las funciones, es momento de crear un menú que nos permita acceder a cada una de ellas de manera sencilla. Esta función mostrar_menu() ofrecerá una interfaz que presenta todas las opciones disponibles y nos guiará en la navegación por las distintas funcionalidades, desde registrar y actualizar hasta buscar, eliminar y generar reportes. Este tipo de menú será también clave para el TFI, ya que organiza las opciones de forma accesible y ayuda a mantener el flujo del programa.

En el menú, cada opción llamará a una de las funciones que hemos implementado, según la elección de la persona que utiliza la aplicación.

```
def mostrar_menu():
    while True:
        print("\nMenú de Gestión de Personas:")
        print("1. Registrar persona")
        print("2. Mostrar personas")
        print("3. Actualizar edad de una persona")
        print("4. Eliminar persona")
```

```
        print("5. Buscar persona")
        print("6. Reporte de personas menores de edad")
        print("7. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            registrar_persona()
        elif opcion == "2":
            mostrar_personas()
        elif opcion == "3":
            actualizar_persona()
        elif opcion == "4":
            eliminar_persona()
        elif opcion == "5":
            buscar_persona()
        elif opcion == "6":
            reporte_menores_edad()
        elif opcion == "7":
            print("Saliendo del programa...")
            break
        else:
            print("Opción inválida. Por favor, intente nuevamente.")
```

Este menú utiliza un *bucle while* que se mantiene activo hasta que la usuaria o usuario elige la opción "7" para salir. Cada vez que se presenta el menú, se selecciona una opción ingresando el número correspondiente. Según la elección, se ejecuta la función respectiva: por ejemplo, al seleccionar "1", se llama a **registrar_persona()** para agregar una nueva persona, mientras que al elegir "6", se muestra el reporte de personas menores de edad con **reporte_menores_edad().**

El bucle es necesario para que el menú reaparezca después de cada operación, permitiendo realizar múltiples acciones sin necesidad de reiniciar el programa. Si se ingresa una opción inválida, se muestra un mensaje de error y se vuelve a mostrar el menú.

Este código de ejemplo te permitirá construir un menú similar para el TFI, adaptado a las funciones que implementaste para gestionar el inventario.

Crear la base de datos y la tabla.

Antes de intentar guardar datos en la tabla Persona, debemos crearla. Ejecuta el siguiente bloque de código una sola vez para crear la tabla en la base de datos:

```
import sqlite3

def crear_tabla_personas():
    # Conectar a la base de datos
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()

    # Crear la tabla Personas si no existe
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS Personas (
            nombre TEXT,
            edad INTEGER,
            ciudad TEXT
        )
    ''')

    # Confirmar la creación de la tabla y cerrar la conexión
    conexion.commit()
    conexion.close()
    print("Tabla Personas creada con éxito.")
```

```
# Llamar a la función para crear la tabla
crear_tabla_personas()
```

En este script, primero nos conectamos a la base de datos base_datos.db. Luego creamos la tabla Personas si no existe ya, con los campos nombre (tipo TEXT), edad (tipo INTEGER), y ciudad (tipo TEXT). Confirmamos los cambios con commit() y cerramos la conexión.

Ejecuta este código una sola vez y la tabla se creará en la base de datos. Luego podrás ejecutar el programa principal sin problemas, ya que la tabla Personas estará disponible para realizar todas las operaciones.

Estructura completa del programa.

Para que el programa funcione en su totalidad, necesitás unir todas las piezas que hemos desarrollado. Esto incluye importar el módulo sqlite3, definir cada función que hemos explicado y luego llamar a la función mostrar_menu() al final del archivo para que el programa comience desde el menú. Vamos a ver cómo quedaría todo el código junto y cómo funciona.

Primero, asegurate de que el archivo contenga las importaciones necesarias, en este caso import sqlite3, que permite trabajar con la base de datos. Luego, debés incluir la definición de cada una de las funciones que desarrollamos para las distintas operaciones.

Al final del archivo, deberás incluir la llamada a mostrar_menu(), que es la función que lanza el menú interactivo del programa.

A continuación podés observar cómo se vería el programa completo, con todas las funciones y el menú integrados:

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```
import sqlite3

# Función para registrar una nueva persona en la base de datos
def registrar_persona():
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    nombre = input("Ingrese el nombre de la persona: ")
    edad = int(input("Ingrese la edad de la persona: "))
    ciudad = input("Ingrese la ciudad de residencia: ")
    cursor.execute("INSERT INTO Personas (nombre, edad, ciudad) VALUES
(?, ?, ?)", (nombre, edad, ciudad))
    conexion.commit()
    print("Persona registrada con éxito.")
    conexion.close()

# Función para mostrar todas las personas registradas en la base de
datos
def mostrar_personas():
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    cursor.execute("SELECT * FROM Personas")
    resultados = cursor.fetchall()
    if resultados:
        print("Listado de personas registradas:")
        for registro in resultados:
            print("Nombre:", registro[0], "Edad:", registro[1],
"Ciudad:", registro[2])
    else:
        print("No hay registros en la tabla Personas.")
    conexion.close()
```

```
# Función para actualizar la edad de una persona específica en la base
de datos
def actualizar_persona():
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    nombre = input("Ingrese el nombre de la persona que desea
actualizar: ")
    nueva_edad = int(input("Ingrese la nueva edad: "))
    cursor.execute("UPDATE Personas SET edad = ? WHERE nombre = ?",
(nueva_edad, nombre))
    if cursor.rowcount > 0:
        print("La edad de", nombre, "ha sido actualizada con éxito.")
    else:
        print("No se encontró a una persona con ese nombre.")
    conexion.commit()
    conexion.close()

# Función para eliminar una persona específica de la base de datos
def eliminar_persona():
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    nombre = input("Ingrese el nombre de la persona que desea eliminar:
")
    cursor.execute("DELETE FROM Personas WHERE nombre = ?", (nombre,))
    if cursor.rowcount > 0:
        print("La persona", nombre, "ha sido eliminada con éxito.")
    else:
        print("No se encontró a una persona con ese nombre.")
    conexion.commit()
    conexion.close()

# Función para buscar una persona en la base de datos
def buscar_persona():
```

```
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    nombre = input("Ingrese el nombre de la persona que desea buscar: ")
    cursor.execute("SELECT * FROM Personas WHERE nombre = ?", (nombre,))
    resultado = cursor.fetchone()
    if resultado:
        print("Información de la persona encontrada:")
        print("Nombre:", resultado[0])
        print("Edad:", resultado[1])
        print("Ciudad:", resultado[2])
    else:
        print("No se encontró a una persona con ese nombre.")
    conexion.close()

# Función para generar un reporte de personas menores de 18 años
def reporte_menores_edad():
    conexion = sqlite3.connect("base_datos.db")
    cursor = conexion.cursor()
    cursor.execute("SELECT * FROM Personas WHERE edad < 18")
    resultados = cursor.fetchall()
    if resultados:
        print("Personas menores de edad:")
        for registro in resultados:
            print("Nombre:", registro[0], "Edad:", registro[1],
"Ciudad:", registro[2])
    else:
        print("No se encontraron personas menores de edad.")
    conexion.close()

# Función del menú principal que permite acceder a cada función
def mostrar_menu():
    while True:
        print("\nMenú de Gestión de Personas:")
```

```
        print("1. Registrar persona")
        print("2. Mostrar personas")
        print("3. Actualizar edad de una persona")
        print("4. Eliminar persona")
        print("5. Buscar persona")
        print("6. Reporte de personas menores de edad")
        print("7. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":
            registrar_persona()
        elif opcion == "2":
            mostrar_personas()
        elif opcion == "3":
            actualizar_persona()
        elif opcion == "4":
            eliminar_persona()
        elif opcion == "5":
            buscar_persona()
        elif opcion == "6":
            reporte_menores_edad()
        elif opcion == "7":
            print("Saliendo del programa...")
            break
        else:
            print("Opción inválida. Por favor, intente nuevamente.")

# Iniciar el programa llamando al menú principal
mostrar_menu()
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

El programa comienza con las importaciones y la definición de cada función. Cuando ejecutas el programa, se llama a mostrar_menu(), que te ofrece las distintas opciones. Podés elegir cualquiera de las funciones presionando el número correspondiente, y luego de completar la operación, el menú vuelve a aparecer, permitiendo realizar tantas operaciones como quieras sin reiniciar el programa.

Cada función maneja su conexión a la base de datos de forma independiente, lo cual garantiza que todas las operaciones (registro, actualización, eliminación, búsqueda y generación de reportes) se realicen correctamente. Este flujo de trabajo también asegura que los datos estén siempre actualizados y accesibles para cualquier consulta.

Este código es adaptable para tu TFI: podés cambiar la estructura de la tabla y los nombres de los campos y ajustar las consultas para que se adapten al inventario de productos que necesitas gestionar. El menú y la organización del programa te darán una estructura sólida para llevar adelante tu proyecto de principio a fin.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Proyecto Final Integrador (PFI).

Estamos en la etapa final del curso y es hora de poner en práctica todo lo que hemos aprendido en el Proyecto Final Integrador (PFI). Este proyecto será la culminación de todo el trabajo que realizaste a lo largo de las clases y es tu oportunidad para aplicar los conocimientos y habilidades que fuiste desarrollando. El objetivo es crear una aplicación en Python capaz de gestionar el inventario de una pequeña tienda. Esta aplicación funcionará desde la terminal y hará uso de una base de datos, permitiendo que los datos se guarden de forma permanente.

La aplicación no solo debe cumplir con las funcionalidades básicas, sino también demostrar tu capacidad para organizar, manipular y presentar información en un sistema real. A través de este proyecto, vas a integrar todo lo que aprendiste sobre estructuras de control, funciones y, ahora, bases de datos. Verás cómo cada elemento cobra sentido cuando desarrollás algo completo y funcional, y además, estarás creando una pieza que podrá formar parte de tu portafolio profesional.

Tu desafío es claro: crear un sistema de inventario que permita registrar productos, consultarlos, actualizarlos, eliminarlos y generar listados y reportes. Cada una de estas funcionalidades te ayudará a consolidar tus conocimientos en Python y en manejo de bases de datos. Además, esta es la oportunidad para demostrar tu capacidad de desarrollar un proyecto de programación completo, manteniendo el código organizado y utilizando las mejores prácticas que hemos discutido.

Objetivos del PFI:

Registro de productos: Tu aplicación deberá permitir ingresar nuevos productos al inventario, solicitando información relevante como nombre, descripción, cantidad disponible, precio y categoría. Ahora que conocés el manejo de bases de datos, podrás almacenar estos datos de manera ordenada y permanente.

Consulta de productos: Tus usuarias y usuarios podrán consultar el inventario y ver la información detallada de cada producto, como stock disponible y precio. La aplicación debe ser clara en la presentación de esta información para que quien la use tenga acceso rápido y preciso a los datos.

Actualización de productos: Si el inventario cambia, tu aplicación deberá reflejar esas actualizaciones. Podrás modificar la cantidad disponible de un producto específico utilizando su ID en la base de datos, lo que garantiza que cada cambio quede correctamente registrado.

Eliminación de productos: La aplicación también debe permitir eliminar productos del inventario en caso de que se dejen de vender, siempre asegurando la integridad del sistema. Esta funcionalidad es clave para manejar el inventario de manera eficiente.

Listado Completo y Reporte de Bajo Stock: Además de consultar productos individualmente, la aplicación deberá generar un listado completo del inventario y un reporte de productos con bajo stock. Esto permitirá tomar decisiones informadas sobre qué productos necesitan ser repuestos.

Este proyecto no solo busca que logres desarrollar una aplicación funcional, sino también trabajar con bases de datos y de estructurar el código en funciones. Al finalizar, tendrás una aplicación completa y funcional que demostrará tus conocimientos y tu capacidad para llevar a cabo un proyecto real. Este PFI no es solo un ejercicio académico: es una experiencia concreta en desarrollo de software que añadirá un gran primer proyecto a tu portafolio profesional.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad