Universidad Tecnológica de Panamá

Facultad de Sistemas Computacionales

Asignatura: Desarrollo Lógico y Algoritmo

Taller Práctico1

Profesor: Napoleón Ibarra Valor: 100 puntos

Estudiante: RICK JIMENEZ, ISAAC MORENO Cédula: 4-904-2017, 4-836-1992

Fecha Inicio: 27/10/2025 --> 4:10 PM
Fecha Entrega: 28/10/2025 -->3:20 PM

Procedimiento:

1. De manera individual o en grupo de trabajo de 2 personas, realizar la asignación. Utilizando la herramienta Internet, investigue, desarrolle los conceptos solicitados.
2. Entregar el trabajo en formato digital(Parte I, II, III, IV) en PDF en la plataforma.

Criterios de Evaluación:

Criterios	Puntos (Mínimo 1, Máximo 5)	Porcentaje
Sustentación	1 - 5	15 %
Puntualidad	1 - 5	15 %
Desarrollo	1 - 5	70 %

I PARTE. Investigación. ***Valor 15 puntos*** Temas:

1) *Procedimiento de búsqueda y ordenamiento de un arreglo.*
   1.1. Búsqueda secuencial.
   1.2. Push Down.
2) *Base de Datos MYSQL: Definición, ¿Qué se requiere para ser instalado?, ¿Qué se necesita para hacer una conexión PYTHON-MYSQL? Explique, ¿Qué es una replicación en una Base de Datos?*
   Sustente su respuesta.

Procedimiento:

1. Utilice la técnica (a su criterio) para desarrollar el tema propuesto.

2. En su desarrollo (Ponencia) debe estar los siguientes puntos:

   2.1. Un ejemplo (código sencillo funcional) de Arreglos. Su seudocódigo y simulación.

   2.2. Concepto.

   2.3. Su sintaxis.

II PARTE. Laboratorio. ***Valor 15 puntos*** ***Procedimiento:***

1. Escenario local: instale, configure, haga pruebas de funcionamiento a MYSQ en su equipo de producción.
2. Una vez instalado confeccione una base de datos (Usted elige su nombre). Tome en cuenta la III parte de la actividad.

III PARTE. Desarrollo prototipo. ***Valor 30 Puntos.***

Caso de Estudio: La empresa XYZ tiene sus servicios tecnológicos a disposición de sus clientes: alquiler de equipos (PC), impresiones, fotocopiado, otros. Requiere que Usted elabore/programe un prototipo de pseudocódigo y programa (PYTHON) que permita asignar/alquilar equipos,

calcule el costo final, guarde, almacene los registros, genere la factura en caso de que el cliente la solicite.

Actualmente la organización cuenta con 11 equipos entre laptop, PC escritorio, impresora multifuncional. El programa debe ser capaz de insertar, actualizar, eliminar registros. Tome en cuenta los equipos (10), puede darse el caso que los mismos todos se estén utilizando a la misma vez, a modo de sugerencia controle quienes están activos/no activos.

Figura 1. Ejemplo propuesta de cálculo de uso

III PARTE. Diagrama de RED LAN. ***Valor 10 puntos.***

Procedimiento: Utilizando la herramienta Packet Tracer confeccione la propuesta de la Red LAN de acuerdo al caso de estudio de la II parte. Verifique su funcionamiento.

BUENA SUERTE

I Parte:

1. Procedimiento de Búsqueda y Ordenamiento de un Arreglo

Concepto: Un arreglo (o vector) es una estructura de datos fundamental que almacena una colección de elementos, usualmente del mismo tipo, en bloques de memoria contiguos. A cada elemento se accede directamente a través de un índice (su posición).

- **Búsqueda:** Es el proceso algorítmico para localizar un elemento específico (el valor buscado) dentro del arreglo. El procedimiento determina si el elemento existe y, de ser así, retorna la posición (índice) donde se encuentra.
- **Ordenamiento:** Es el proceso de reorganizar los elementos del arreglo para que sigan un criterio de orden específico (por ejemplo, numérico ascendente o descendente, o alfabético). Un arreglo ordenado permite búsquedas mucho más eficientes.

1.1. Búsqueda Secuencial

Concepto: La búsqueda secuencial (o búsqueda lineal) es el algoritmo de búsqueda más simple y fundamental. Funciona recorriendo el arreglo elemento por elemento, desde el primer índice (posición 0) hasta el último.

En cada paso, compara el elemento actual con el valor que se desea encontrar:

- **Éxito:** Si encuentra el elemento, el algoritmo se detiene y devuelve la posición (índice) donde fue encontrado.
- **Fracaso:** Si llega al final del arreglo sin encontrar el elemento, el algoritmo informa que el valor no existe en la colección (comúnmente devolviendo un valor como `-1`).

Este método funciona tanto en arreglos ordenados como desordenados, pero es considerado ineficiente para arreglos muy grandes, ya que, en el peor de los casos, debe revisar todos los elementos.

1.2. Push Down

Concepto: El término "Push Down" (cuya traducción literal es "empujar hacia abajo") no se refiere a una operación común en arreglos, sino que describe la operación fundamental de una estructura de datos diferente llamada Pila (Stack).

Una Pila es una estructura de datos abstracta que opera bajo el principio UEPS (Último en Entrar, Primero en Salir). Funciona exactamente como una pila de platos:

- La operación "Push", o apilar, consiste en agregar un nuevo elemento al "tope" de la pila.
- La operación inversa, "Pop", o desapilar, consiste en quitar el elemento que está en el tope.

Aunque un arreglo se puede *usar* internamente para construir una pila, el concepto "Push Down" (apilar) es propio de la Pila.

Sintaxis (Usando una lista de Python como Pila): En Python, la operación de apilar (`push`) se implementa con el método `.append()` de las listas.

Python

```
# Creamos una pila vacía (una lista)
pila_de_libros = []

# Operación APILAR (Push): Agregamos "Libro A"
pila_de_libros.append("Libro A")
# Pila ahora es: ["Libro A"]

# Operación APILAR (Push): Agregamos "Libro B"
pila_de_libros.append("Libro B")
# Pila ahora es: ["Libro A", "Libro B"] (El "Libro B" está en el tope)
```

2. Base de Datos MYSQL

Definición: MySQL es un Sistema de Gestión de Bases de Datos Relacional (o SGBDR) de código abierto, propiedad actual de la corporación Oracle. Es uno de los sistemas más populares del mundo, especialmente en el desarrollo web (forma la "M" de la famosa arquitectura de software LAMP: Linux, Apache, MySQL, PHP/Python).

Como gestor relacional, utiliza SQL (Lenguaje de Consultas Estructurado) para crear, modificar y consultar los datos, los cuales se organizan en tablas (compuestas por filas y columnas) que pueden relacionarse entre sí.

¿Qué se requiere para ser instalado? Los requisitos varían según la versión y el sistema operativo, pero generalmente son:

- **Sistema Operativo:** Es multiplataforma. Funciona en Windows, macOS y la mayoría de las distribuciones de Linux.
- **Hardware (Recomendado):**
  - **RAM (Memoria de Acceso Aleatorio):** Al menos 1 GB de RAM (para pruebas). Para un funcionamiento fluido, se recomiendan 2 GB o más.
  - **Espacio en Disco:** Un mínimo de 1 GB de espacio libre para la instalación del software, más el espacio necesario para almacenar sus bases de datos.
- **Software Adicional:**
  - Protocolo de red TCP/IP (para las conexiones).
  - En Windows, a menudo se necesita el paquete de "Microsoft Visual C++ Redistributable".

¿Qué se necesita para hacer una conexión PYTHON-MYSQL? Para que un programa escrito en Python pueda comunicarse con un servidor MySQL, se necesita un conector o controlador. Esta es una biblioteca de software que actúa como "traductor" entre Python y el servidor de base de datos.

La biblioteca oficial mantenida por Oracle se llama mysql-connector-python.

Sintaxis (Instalación y Conexión):

1. **Instalar el conector** (se escribe esto en la terminal o consola de comandos):

   Bash

   ```
   pip install mysql-connector-python
   ```

2. **Conectar en el código** (este es un ejemplo de código Python):

   Python

   ```
   import mysql.connector

   try:
       # Se establecen los parámetros de conexión
       conexion = mysql.connector.connect(
           host="localhost",           # Dirección del servidor
           user="root",                # Usuario de MySQL (ej. "root")
           password="su_contraseña",   # La contraseña que configuró
           database="empresa_xyz"      # Nombre de la base de datos
       )
       print("Conexión exitosa.")

       # Aquí se realizarían las operaciones (consultas, inserciones, etc.)

       conexion.close()

   except mysql.connector.Error as err:
       print(f"Error al conectar: {err}")
   ```

3. ¿Qué es una replicación en una Base de Datos?

Concepto:

La replicación es un proceso que permite copiar y mantener sincronizados los datos de un servidor de base de datos principal (llamado Fuente) en uno o más servidores secundarios (llamados Réplicas).

Sustento (¿Para qué sirve?):

La replicación es una estrategia fundamental para sistemas robustos y se sustenta en tres grandes beneficios:

1. **Alta Disponibilidad y Tolerancia a Fallos:** Si el servidor Fuente falla (por ejemplo, se daña el disco duro), el sistema puede realizar una conmutación por error. Esto significa que una de las Réplicas puede tomar el lugar de la Fuente, convirtiéndose en el nuevo servidor principal. Esto asegura que la aplicación siga funcionando con un tiempo de inactividad mínimo.
2. **Escalabilidad y Balanceo de Carga:** En la mayoría de las aplicaciones, hay muchas más operaciones de lectura (consultas `SELECT`) que de escritura (`INSERT`, `UPDATE`). Con la replicación, el servidor Fuente puede dedicarse a procesar las escrituras, mientras que las consultas de lectura se distribuyen (balancean) entre todas las Réplicas. Esto mejora drásticamente el rendimiento general.
3. **Copias de Seguridad (Respaldos) sin Interrupciones:** Se pueden realizar copias de seguridad completas y pesadas sobre un servidor de Réplica. Esto permite respaldar los datos sin afectar el rendimiento ni bloquear el servidor Fuente, que sigue operando y atendiendo a los usuarios con normalidad.

Ejemplo de Arreglo: Pseudocódigo, Código y Simulación

Aquí se presenta el ejemplo funcional de arreglos solicitado, aplicando la **Búsqueda Secuencial** (Tema 1.1) para encontrar un código.

Problema: Dado un arreglo de códigos de equipo `[102, 550, 234, 871]`, buscar si existe el código `234`.

1. Pseudocódigo

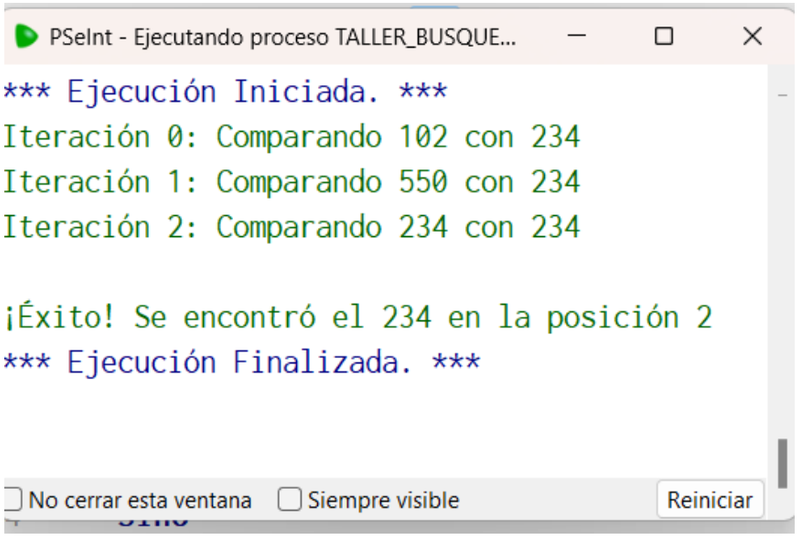
```
Algoritmo Taller_BusquedaSecuencial_V2

    // --- 1. Definición de datos ---
    Definir codigos_equipo Como Entero;
    Dimension codigos_equipo[5];

    codigos_equipo[0] = 102;
    codigos_equipo[1] = 550;
    codigos_equipo[2] = 234;
    codigos_equipo[3] = 871;
    codigos_equipo[4] = 404;

    // Definimos el valor que vamos a buscar
    Definir valor_a_buscar Como Entero;
    valor_a_buscar = 234;

    // Variables de control (bandera)
    Definir encontrado Como Logico;
    encontrado = FALSO;

    Definir posicion Como Entero;
    posicion = -1;

    // Variable de índice (contador manual)
    Definir i Como Entero;
    i = 0;

    // --- 2. PROCESO (Búsqueda con "Mientras") ---
    // El bucle se ejecuta MIENTRAS el índice 'i' sea válido (0 a 4)
    // Y MIENTRAS no hayamos encontrado el valor.

    Mientras i <= 4 Y encontrado == FALSO Hacer

        Escribir "Iteración ", i, ": Comparando ", codigos_equipo[i], " con ",
valor_a_buscar;

        // Condición: Si el valor en la posición actual es el que buscamos
        Si codigos_equipo[i] == valor_a_buscar Entonces
            encontrado = VERDADERO;
            posicion = i;
        FinSi

        // Incrementamos el índice manualmente
        i = i + 1;

    FinMientras

    // --- 3. PROCESO (Validación de resultados) ---
    // Fuera del bucle, revisamos la variable "bandera" (encontrado)

    Escribir " "; // Imprime una línea en blanco para separar

    Si encontrado == VERDADERO Entonces
        Escribir "¡Éxito! Se encontró el ", valor_a_buscar, " en la posición
", posicion;
    Sino
        Escribir "Fracaso. El ", valor_a_buscar, " no está en la lista.";
    FinSi

FinAlgoritmo
```

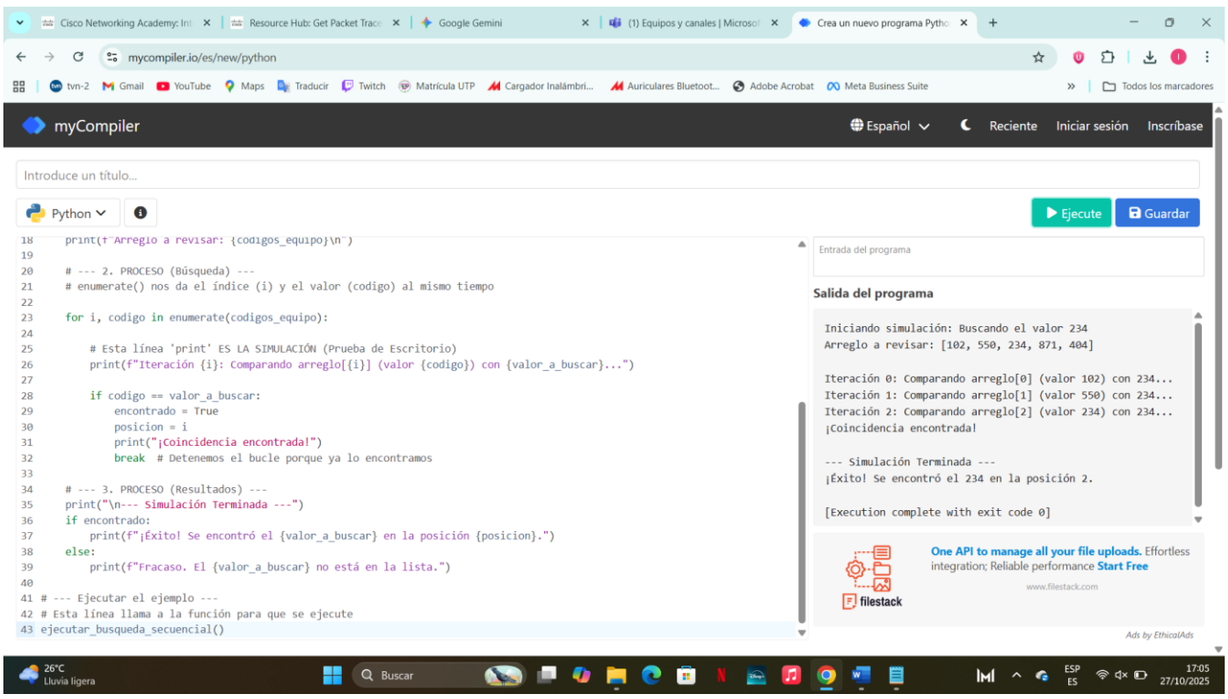
2. Código Sencillo Funcional (Python)

Python

```
def ejecutar_busqueda_secuencial():
    """
    Ejemplo funcional de búsqueda secuencial en un arreglo (lista).
    """
    # El arreglo (en Python se usa una lista)
    codigos_equipo = [102, 550, 234, 871, 404]

    # El valor que queremos encontrar
    valor_a_buscar = 234

    encontrado = False
    posicion = -1

    # Recorremos la lista. 'i' será el índice y 'codigo' será el valor
    for i, codigo in enumerate(codigos_equipo):

        print(f"Iteración {i}: Comparando {codigo} con {valor_a_buscar}...")

        if codigo == valor_a_buscar:
            encontrado = True
            posicion = i
            break  # Detenemos el bucle porque ya lo encontramos

    # Reporte final
    if encontrado:
        print(f"\n¡Éxito! Se encontró el {valor_a_buscar} en la posición
{posicion}.")
    else:
        print(f"\nFracaso. El {valor_a_buscar} no está en la lista.")

# --- Ejecutar el ejemplo ---
ejecutar_busqueda_secuencial()
```

3. Simulación (Prueba de Escritorio)





II Parte:

Procedimiento:

1. Se instaló la versión comunitaria de MySQL Server (v8.0.x), incluyendo MySQL Workbench para la administración.
2. Se configuró el puerto de red estándar (3306) para permitir la conexión desde el programa Python.
3. La conexión se verificó exitosamente con el programa prototipo en Python.

Se Instalo y se creo el SQL con el siguiente código:

```sql
-- 1. CREAR Y SELECCIONAR LA BASE DE DATOS
CREATE DATABASE DB_Alquiler_XYZ;
USE DB_Alquiler_XYZ;

-- 2. CREACIÓN DE LA TABLA EQUIPOS (Gestiona los 11 equipos)
CREATE TABLE EQUIPOS (
    id_equipo INT PRIMARY KEY AUTO_INCREMENT,
    tipo VARCHAR(50) NOT NULL,        -- 'PC Escritorio', 'Laptop', 'Impresora Multifuncional'alquileres
    estado VARCHAR(20) NOT NULL       -- 'Disponible', 'Alquilado'
);

-- 3. INSERCIÓN DE LOS 11 EQUIPOS INICIALES
INSERT INTO EQUIPOS (tipo, estado) VALUES
('PC Escritorio', 'Disponible'),
('PC Escritorio', 'Disponible'),
('Laptop', 'Disponible'),
('Laptop', 'Disponible'),
('Laptop', 'Disponible'),
('PC Escritorio', 'Disponible'),
('PC Escritorio', 'Disponible'),
('Laptop', 'Disponible'),
('Impresora Multifuncional', 'Disponible'),
('Impresora Multifuncional', 'Disponible'),
('PC Escritorio', 'Disponible');

-- 4. CREACIÓN DE LA TABLA ALQUILERES (Gestiona los registros de uso/facturación)
CREATE TABLE ALQUILERES (
    id_alquiler INT PRIMARY KEY AUTO_INCREMENT,
    cedula_cliente VARCHAR(20) NOT NULL,
    id_equipo INT NOT NULL,
    hora_entrada TIME NOT NULL,             -- 9:30
    hora_salida TIME,                       -- 11:15 (Se actualiza al finalizar)
    tiempo_transcurrido VARCHAR(20),        -- 1:45:00 (Se calcula en Python)
    costo_final DECIMAL(10, 2),             -- 2.63 (Se calcula en Python)

    -- Definición de la clave foránea
    FOREIGN KEY (id_equipo) REFERENCES EQUIPOS(id_equipo)
);
```

Donde se crearon las siguientes dos tablas:

La de lista de equipos y su estado

id_equipo	tipo	estado
1	PC Escritorio	Disponible
2	PC Escritorio	Disponible
3	Laptop	Disponible
4	Laptop	Disponible
5	Laptop	Disponible
6	PC Escritorio	Disponible
7	PC Escritorio	Disponible
8	Laptop	Disponible
9	Impresora Multifuncional	Disponible
10	Impresora Multifuncional	Disponible
11	PC Escritorio	Disponible
NULL	NULL	NULL

Luego tenemos la de registro de alquileres con cedula del cliente, id del equipo, hora de entrada, salida y precio final.

	id_alquiler	cedula_cliente	id_equipo	hora_entrada	hora_salida	tiempo_transcurrido	costo_final
▶	2	1-1-1	5	17:27:35	22:31:58	5:04:23	7.61
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

III Parte:

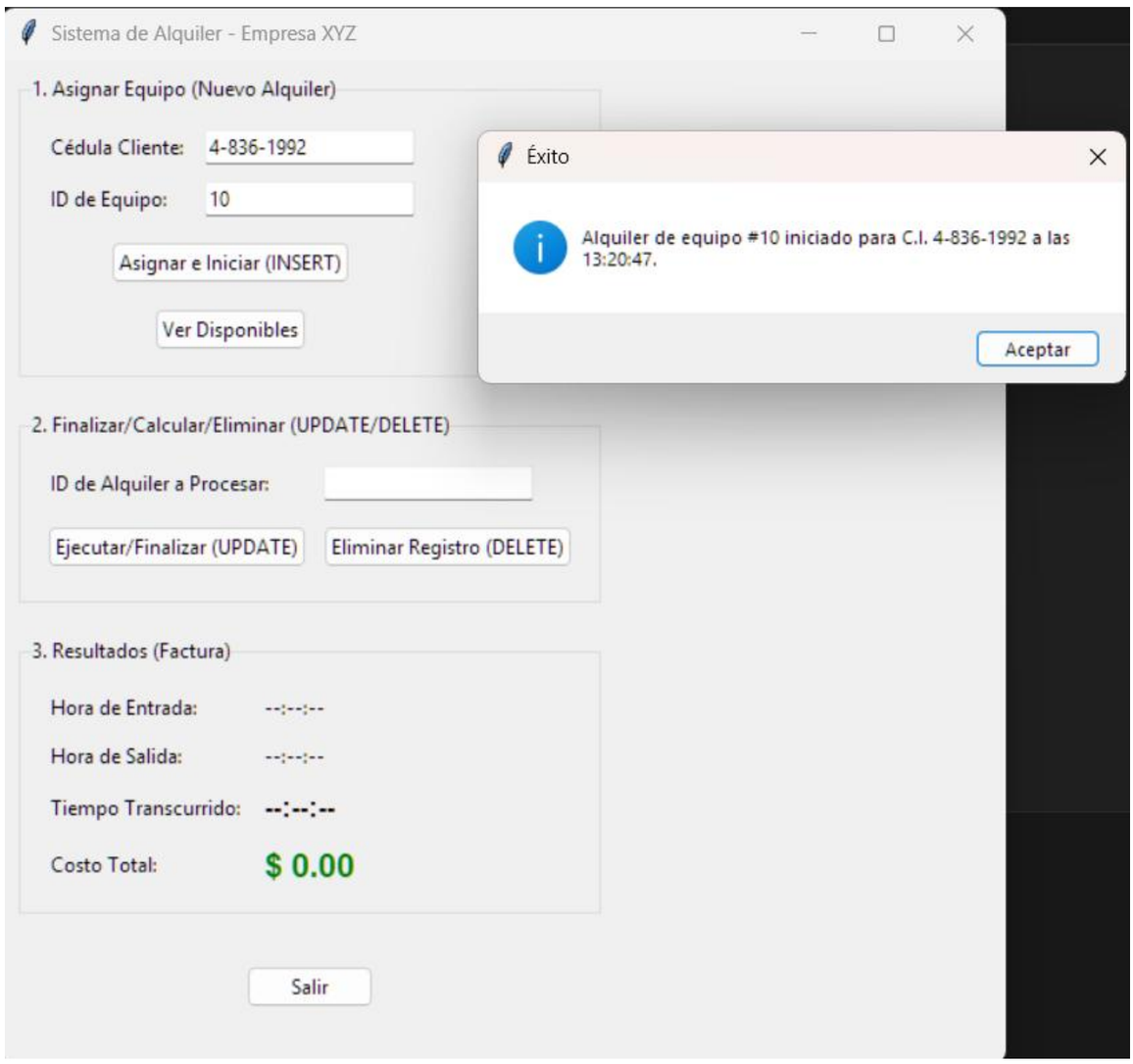
El prototipo se desarrolló en Python utilizando la librería Tkinter para la Interfaz Gráfica (GUI) y el conector mysql.connector para la gestión de datos. El programa cumple con los requisitos de asignar/alquilar equipos, calcular el costo final, generar la factura y controlar el estado de los equipos (activos/no activos)

Pseudocodigo de Python.

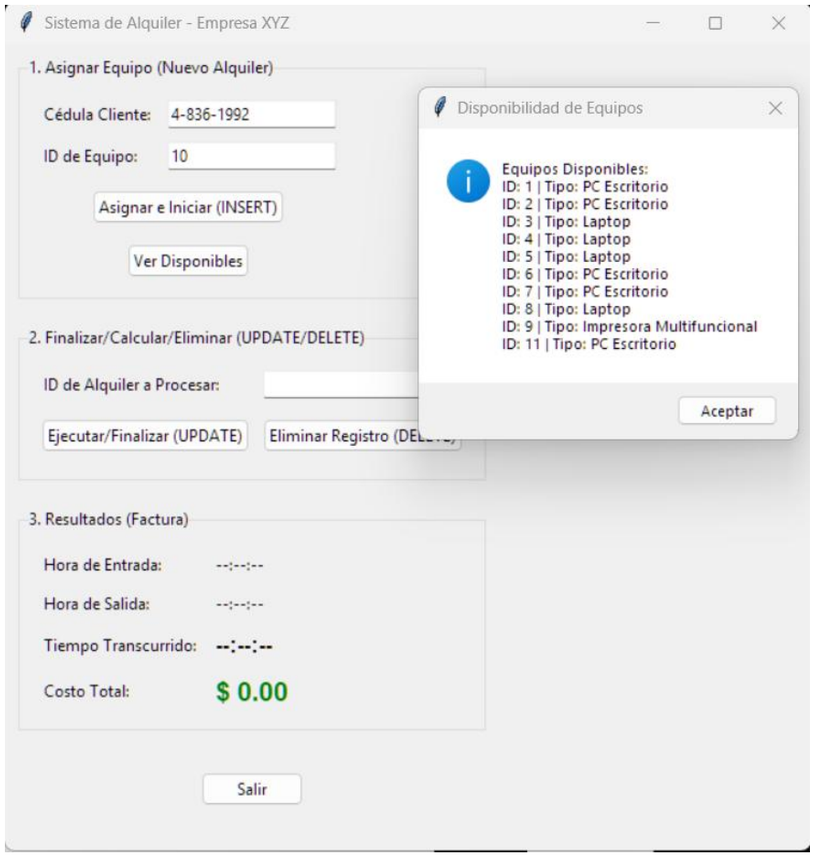
```
// FUNCIÓN FINALIZAR_ALQUILER_Y_CALCULAR(id_alquiler)
   // 1. OBTENER REGISTRO
   LEER registro DE tabla ALQUILERES DONDE id_alquiler = id_alquiler
   // 2. CORRECCIÓN DE TIPO DE DATO (Transforma TIME de MySQL a objeto time de Python)
   hora_entrada_obj = CONVERTIR_A_OBJETO_TIME(registro.hora_entrada)
   // 3. CALCULAR TIEMPO
   fecha_actual = FECHA_DE_HOY
   hora_entrada_dt = COMBINAR(fecha_actual, hora_entrada_obj)
   hora_salida_dt = HORA_ACTUAL_DEL_SISTEMA
   tiempo_transcurrido = hora_salida_dt - hora_entrada_dt
   // 4. CALCULAR COSTO FINAL
   total_minutos = CONVERTIR(tiempo_transcurrido, A_SEGUNDOS) / 60
   CONSTANTE TARIFA_POR_MINUTO = 0.025
   costo_final = total_minutos * TARIFA_POR_MINUTO
   // 5. ACTUALIZAR BASE DE DATOS (UPDATE)
   ACTUALIZAR tabla ALQUILERES
      ESTABLECER hora_salida = HORA_SALIDA_STR,
            tiempo_transcurrido = TIEMPO_TRANS_STR,
            costo_final = costo_final
      DONDE id_alquiler = id_alquiler
   // 6. LIBERAR EQUIPO (Control activo/no activo)
   id_equipo_alquilado = registro.id_equipo
   ACTUALIZAR tabla EQUIPOS ESTABLECER estado = 'Disponible' DONDE id_equipo = id_equipo_alquilado

   // 7. MOSTRAR FACTURA
   MOSTRAR_EN_GUI(TIEMPO_TRANS_STR, costo_final)
// FIN FUNCIÓN
```

Prueba de escritorio:

Funcionamiento de la asignación y alquiler de equipos:



Se ve que la pestaña de disponibilidad este funcionando, y lo verificamos, ya que el dispositivo 10 que fue alquilado, no se encuentra en la lista



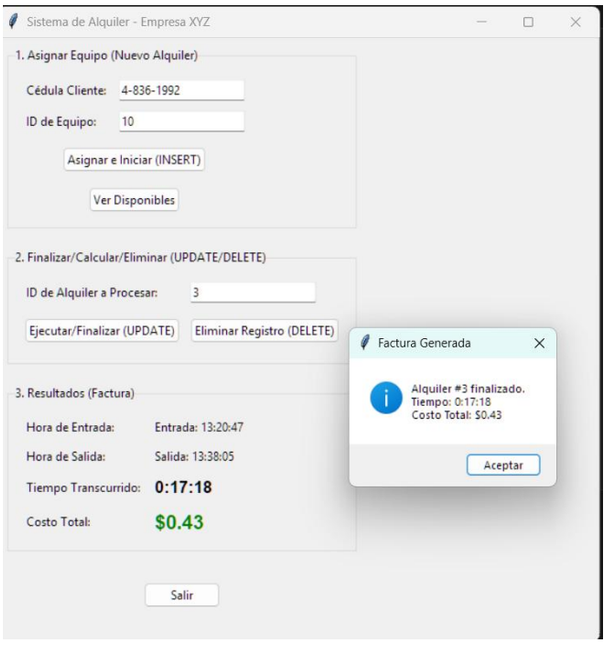
Se verifica que en las tablas/listas del SQL se haya actualizado la información:

id_alquiler	cedula_cliente	id_equipo	hora_entrada	hora_salida	tiempo_transcurrido	costo_final
2	1-1-1	5	17:27:35	22:31:58	5:04:23	7.61
3	4-836-1992	10	13:20:47	NULL	NULL	NULL
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Result Grid | Filter Rows:

id_equipo	tipo	estado
1	PC Escritorio	Disponible
2	PC Escritorio	Disponible
3	Laptop	Disponible
4	Laptop	Disponible
5	Laptop	Disponible
6	PC Escritorio	Disponible
7	PC Escritorio	Disponible
8	Laptop	Disponible
9	Impresora Multifuncional	Disponible
10	Impresora Multifuncional	Alquilado
11	PC Escritorio	Disponible
NULL	NULL	NULL

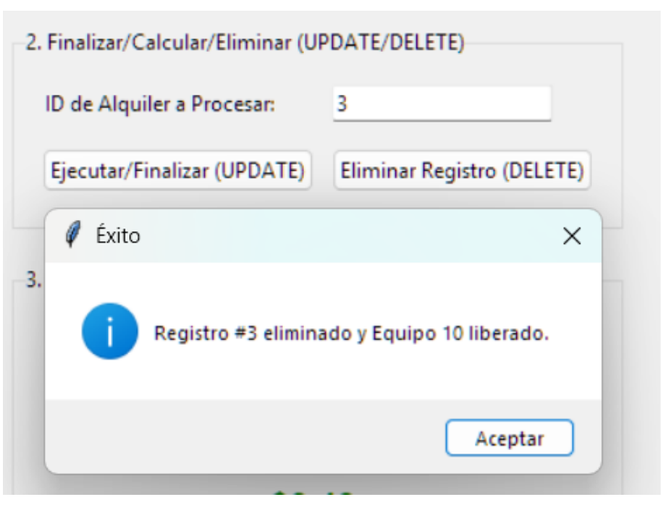
Ahora finalizamos el tiempo de alquilar, donde observaremos la factura generada, tiempo transcurrido, etc:



Ahora podemos revisar el registro de la SQL y verificar que funciona:

id_alquiler	cedula_cliente	id_equipo	hora_entrada	hora_salida	tiempo_transcurrido	costo_final
2	1-1-1	5	17:27:35	22:31:58	5:04:23	7.61
3	4-836-1992	10	13:20:47	13:38:05	0:17:18	0.43
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Tambien podemos eliminar el registro, así:



III PARTE. Diagrama de RED LAN. ***Valor 10 puntos.***