**Universidad Tecnológica de Panamá**

**Facultad de Sistemas Computacionales**

**Asignatura: Desarrollo Lógico y Algoritmo**

**Taller Práctico1**

**Profesor: Napoleón Ibarra** **Valor: 100 puntos**

**Estudiante: Eliana Martínez** **Cédula: 4-834-1382**

**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.

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

1.1. Búsqueda secuencial.

**Concepto:**

La búsqueda secuencial (o lineal) consiste en recorrer un arreglo elemento por elemento hasta encontrar el valor buscado o llegar al final del arreglo. Es el método más simple y no requiere que los datos estén ordenados.

**Sintaxis General:**

| Ejemplo Funcional | Seudocódigo |
|---|---|
| `def busqueda_secuencial(lista, valor):`<br>`  for i in range(len(lista)):`<br>`  if lista[i] == valor:`<br>`  return i  # Retorna la posición donde se encontró`<br>`  return -1  # Si no se encuentra, devuelve -1`<br><br>`# Lista predeterminada`<br>`arreglo = [10, 23, 5, 7, 15, 30]`<br><br>`# Preguntar al usuario si quiere ingresar su propio arreglo`<br>`respuesta = input("¿Quieres ingresar tu propio arreglo? (s/n): ").lower()`<br><br>`if respuesta == "s":`<br>`  arreglo = []`<br>`  cantidad = int(input("¿Cuántos números tendrá tu arreglo? "))`<br>`  for i in range(cantidad):`<br>`  while True:`<br>`  try:`<br>`  num = int(input(f"Ingrese el número {i+1}: "))`<br>`  arreglo.append(num)`<br>`  break`<br>`  except ValueError:`<br>`  print("Por favor, ingresa un número válido.")`<br>`else:`<br>`  print("Se usará el arreglo predeterminado:", arreglo)`<br><br>`# Pedir número a buscar`<br>`while True:`<br>`  try:` | Proceso BusquedaSecuencial<br>  Definir numeros Como Entero[]<br>  Definir cantidad, i, valor, posicion, num Como Entero<br>  Definir respuesta Como Caracter<br><br>  // Arreglo predeterminado<br>  numeros <- [10, 23, 5, 7, 15, 30]<br><br>  Escribir "¿Quieres ingresar tu propio arreglo? (s/n): "<br>  Leer respuesta<br><br>  Si respuesta = "s" Entonces<br>  numeros <- []<br>  Repetir<br>  Escribir "¿Cuántos números tendrá tu arreglo? "<br>  Leer cantidad<br>  Si cantidad <= 0 Entonces<br>  Escribir "Ingresa un número mayor que 0."<br>  FinSi<br>  Hasta Que cantidad > 0<br><br>  Para i <- 1 Hasta cantidad<br>  Repetir<br>  Escribir "Ingresa el número ", i, ": "<br>  Leer num<br>  Hasta Que EsEntero(num)<br>  numeros[i] <- num<br>  FinPara<br>  Sino<br>  Escribir "Se usará el arreglo predeterminado: ", numeros<br>  FinSi<br><br>  Escribir "Ingrese el número que desea buscar: "<br>  Leer valor |

```
        valor = int(input("Ingrese el
número que desea buscar: "))
        break
    except ValueError:
        print("Por favor, ingresa un número
válido.")

# Realizar búsqueda
pos = busqueda_secuencial(arreglo, valor)

# Mostrar resultado
if pos != -1:
    print(f"El número {valor} se encuentra
en la posición {pos}.")
else:
    print(f"El número {valor} no se
encuentra en el arreglo.")
```

```
// Búsqueda secuencial
posicion <- -1
Para i <- 1 Hasta Longitud(numeros)
   Si numeros[i] = valor Entonces
      posicion <- i
      Salir
   FinSi
FinPara

// Mostrar resultado
Si posicion != -1 Entonces
   Escribir "El número ", valor, " se encuentra
en la posición ", posicion
Sino
   Escribir "El número ", valor, " no se
encuentra en el arreglo"
FinSi
FinProceso
```

ejemplo funcional.py > ...

```
1  def busqueda_secuencial(lista, valor):
2      for i in range(len(lista)):
3          if lista[i] == valor:
4              return i  # Retorna la posición donde se encontró
5      return -1  # Si no se encuentra, devuelve -1
6
7  # Lista predeterminada
8  arreglo = [10, 23, 5, 7, 15, 30]
9
10 # Preguntar al usuario si quiere ingresar su propio arreglo
11 respuesta = input("¿Quieres ingresar tu propio arreglo? (s/n): ").lower()
12
13 if respuesta == "s":
14     arreglo = []
15     cantidad = int(input("¿Cuántos números tendrá tu arreglo? "))
16     for i in range(cantidad):
17         while True:
18             try:
19                 num = int(input(f"Ingrese el número {i+1}: "))
20                 arreglo.append(num)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS pwsh

```
PS C:\Users\Usuario\Downloads\Python> & C:\Users\Usuario\AppData\Local\Programs\Python\Python3
14\python.exe "c:/Users/Usuario/Downloads/Python/ejemplo funcional.py"
¿Quieres ingresar tu propio arreglo? (s/n): n
Se usará el arreglo predeterminado: [10, 23, 5, 7, 15, 30]
Ingrese el número que desea buscar: 7
El número 7 se encuentra en la posición 3.
PS C:\Users\Usuario\Downloads\Python>
```

1.2. Push Down.

**Concepto:**

El algoritmo Push Down o Bubble Sort compara elementos adyacentes de un arreglo e intercambia sus posiciones si están en el orden incorrecto.

Con cada pasada, el elemento más grande "baja" (push down) hacia el final del arreglo.

**Sintaxis General:**

Ejemplo Funcional

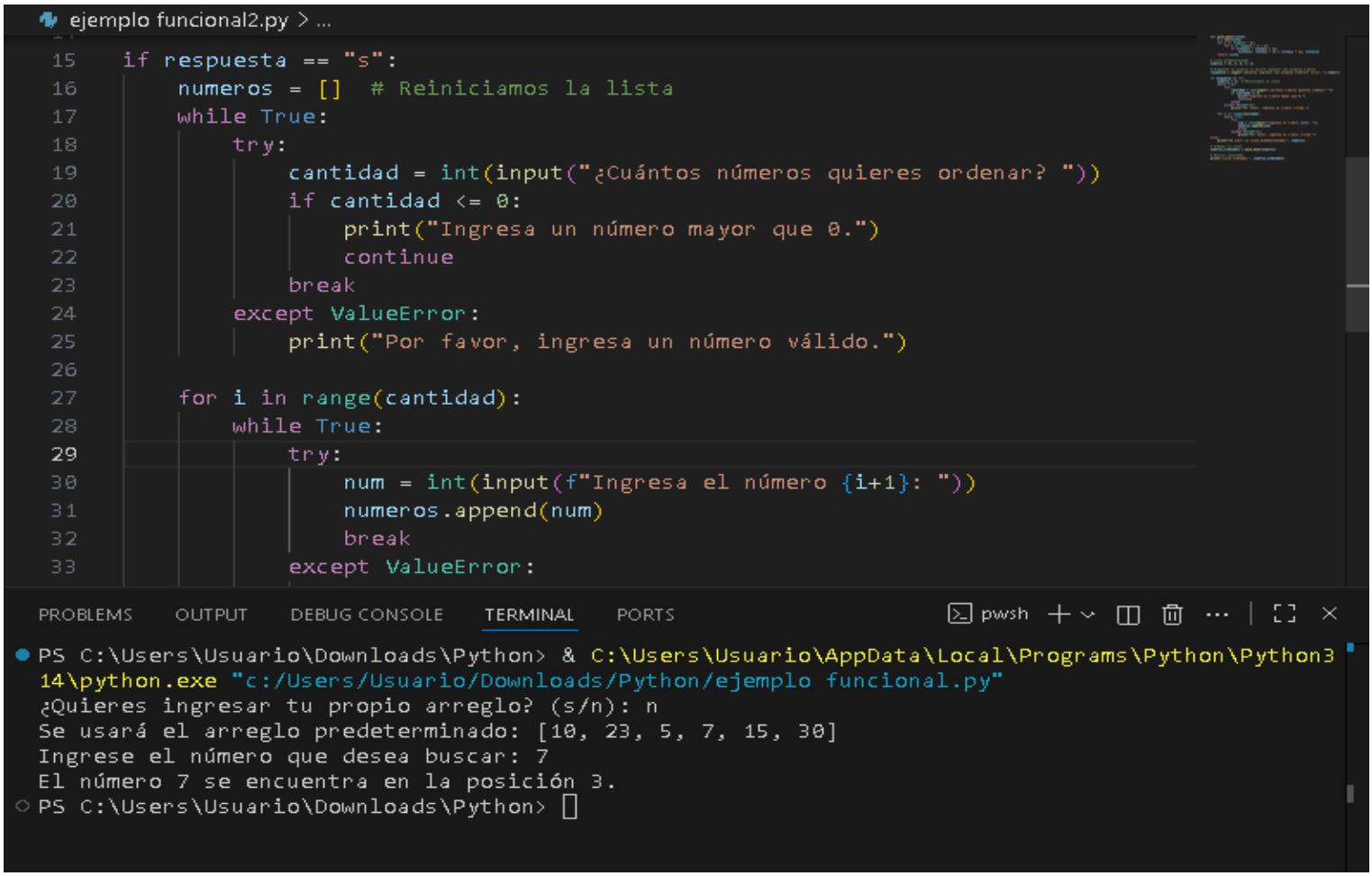
```python
def push_down(lista):
    n = len(lista)
    for i in range(n - 1):
        for j in range(n - 1 - i):
            if lista[j] > lista[j + 1]:
                lista[j], lista[j + 1] =
lista[j + 1], lista[j]
    return lista

# Lista predeterminada
numeros = [5, 2, 8, 3, 1]

# Preguntar al usuario si quiere ingresar sus
propios números
respuesta = input("¿Quieres ingresar tus
propios números? (s/n): ").lower()

if respuesta == "s":
    numeros = []  # Reiniciamos la lista
    while True:
        try:
            cantidad = int(input("¿Cuántos
números quieres ordenar? "))
            if cantidad <= 0:
                print("Ingresa un número mayor
que 0.")
                continue
            break
        except ValueError:
            print("Por favor, ingresa un número
válido.")

    for i in range(cantidad):
        while True:
            try:
                num = int(input(f"Ingresa el
número {i+1}: "))
```

Seudocódigo

```
Proceso OrdenarNumeros
   Definir numeros Como Entero[]
   Definir respuesta, num Como Entero
   Definir cantidad, i, j, n Como Entero

   // Lista predeterminada
   numeros <- [5, 2, 8, 3, 1]

   Escribir "¿Quieres ingresar tus propios números?
(s/n): "
   Leer respuesta

   Si respuesta = "s" Entonces
      numeros <- []

      Repetir
         Escribir "¿Cuántos números quieres ordenar? "
         Leer cantidad
         Si cantidad <= 0 Entonces
            Escribir "Ingresa un número mayor que 0."
         FinSi
      Hasta Que cantidad > 0

      Para i <- 1 Hasta cantidad
         Repetir
            Escribir "Ingresa el número ", i, ": "
            Leer num
            Si EsEntero(num) Entonces
               numeros[i] <- num
               Salir
            Sino
               Escribir "Por favor, ingresa un número
válido."
            FinSi
         FinRepetir
      FinPara
   Sino
```

```
                numeros.append(num)
                break
            except ValueError:
                print("Por favor, ingresa un
número válido.")
else:
    print("Se usará la lista predeterminada:",
numeros)

# Ordenar la lista
numeros_ordenados = push_down(numeros)

# Mostrar resultado
print("Lista ordenada:", numeros_ordenados)
```

```
    Escribir "Se usará la lista predeterminada: ",
numeros
  FinSi

  // Ordenamiento burbuja
  n <- Longitud(numeros)
  Para i <- 1 Hasta n-1
    Para j <- 1 Hasta n-i
      Si numeros[j] > numeros[j+1] Entonces
        num <- numeros[j]
        numeros[j] <- numeros[j+1]
        numeros[j+1] <- num
      FinSi
    FinPara
  FinPara

  Escribir "Lista ordenada: ", numeros
FinProceso
```



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.

**Concepto: MySQL** es un sistema de gestión de bases de datos relacional (RDBMS, por sus siglas en inglés) basado en el lenguaje **SQL (Structured Query Language)**. Permite almacenar, organizar y gestionar grandes volúmenes de información mediante tablas relacionadas entre sí.
Es de código abierto, gratuito en su versión comunitaria y ampliamente usado en aplicaciones web, servidores y proyectos empresariales.

**Características principales:**

- Multiplataforma (funciona en Windows, Linux, macOS, etc.).
- Compatible con lenguajes como Python, PHP, Java, C++, entre otros.
- Rápido, estable y con buena seguridad.
- Permite conexiones remotas y replicación de datos.

## ¿Qué se requiere para ser instalado?

Para instalar MySQL en un equipo se necesita lo siguiente:

Requisitos básicos:

- Sistema operativo compatible: Windows, Linux o macOS.
- Procesador: mínimo 1 GHz (recomendado 2 GHz o más).
- Memoria RAM: mínimo 1 GB (recomendado 2 GB o más).
- Espacio en disco: alrededor de 500 MB o más según el tamaño de las bases de datos.

Software necesario:

- Instalador de MySQL (disponible en https://dev.mysql.com/downloads/).
- Durante la instalación se configuran componentes como:
    - MySQL Server: motor principal de la base de datos.
    - MySQL Workbench: interfaz gráfica para administrar bases de datos.
    - MySQL Shell o Command Line Client: herramientas para ejecutar comandos SQL.

**¿Qué se necesita para hacer una conexión Python–MySQL?**
Para conectar Python con MySQL, se utiliza la librería mysql.connector.
Se requiere tener instalado **MySQL Server**, crear una base de datos y luego establecer la conexión desde Python.

**Ejemplo de conexión:**

```
import mysql.connector

conexion = mysql.connector.connect(
```

```
    host="localhost",
    user="root",
    password="tu_contraseña",
    database="empresa_xyz"
)

if conexion.is_connected():
    print("Conexión exitosa a la base de datos MySQL.")
```

**Explicación:**

- host: dirección del servidor (por defecto “localhost”).
- user: nombre de usuario de MySQL (normalmente “root”).
- password: contraseña del usuario.
- database: nombre de la base de datos a la que se conectará.

**¿Qué es una replicación en una base de datos?**

La replicación en MySQL es el proceso de copiar los datos de un servidor principal (maestro) a uno o varios servidores secundarios (esclavos) de forma automática y continua. Sirve para mantener varias copias actualizadas de una base de datos.

**Tipos de replicación:**

- **Maestro–Esclavo:** el maestro realiza las escrituras y los esclavos solo lecturas.
- **Multimaestro:** varios servidores pueden escribir y sincronizarse entre sí.

**Ventajas:**

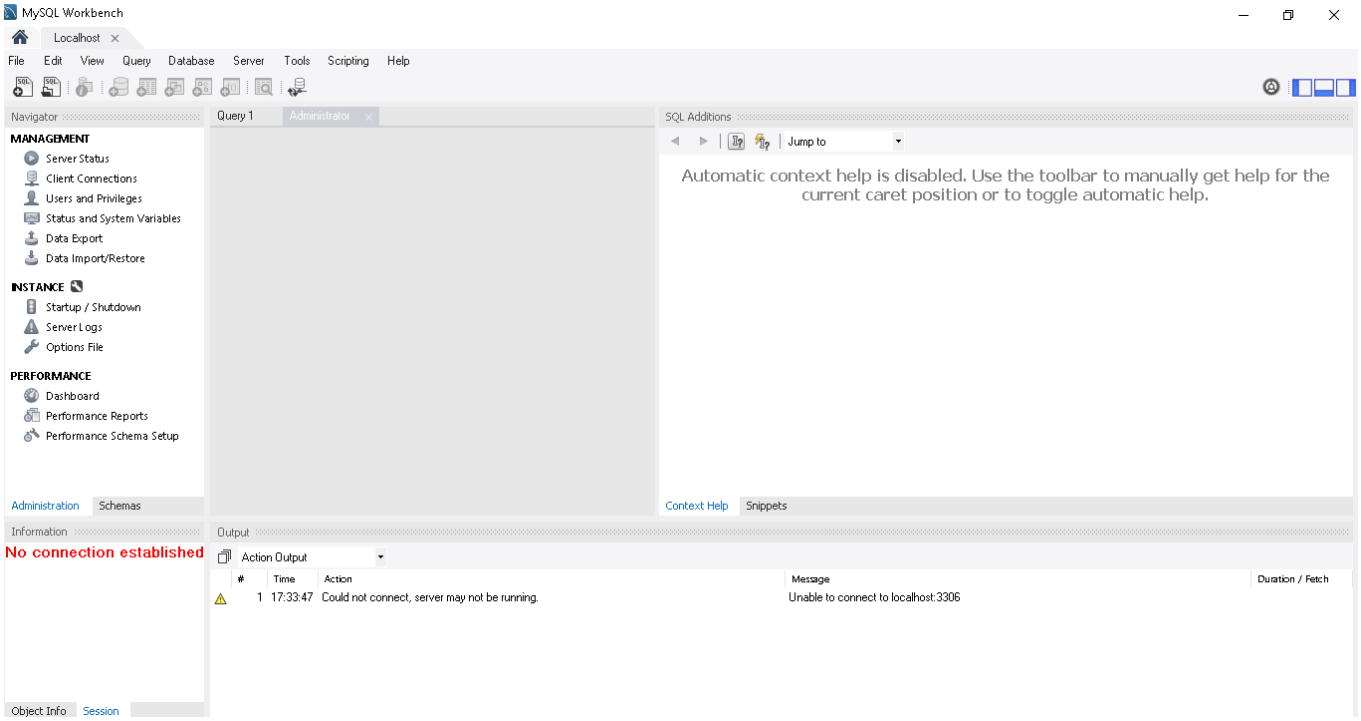
- Mayor disponibilidad del servicio (si un servidor falla, otro puede continuar).
- Mejor rendimiento al distribuir las consultas.
- Copias de respaldo automáticas.

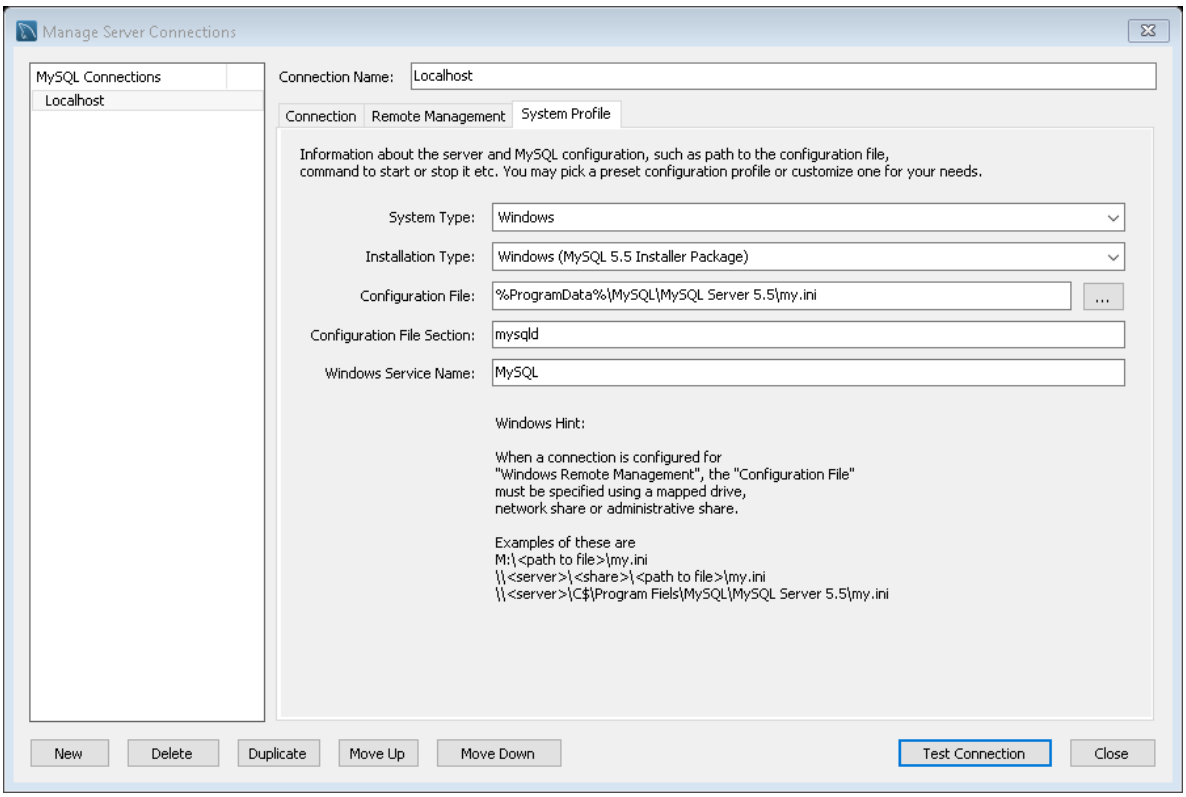
II PARTE. Laboratorio. Valor 15 puntos

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.


MySQL Workbench
Localhost
File Edit View Query Database Server Tools Scripting Help
Navigator
MANAGEMENT
Server Status
Client Connections
Users and Privileges
Status and System Variables
Data Export
Data Import/Restore
INSTANCE
Startup / Shutdown
Server Logs
Options File
PERFORMANCE
Dashboard
Performance Reports
Performance Schema Setup
Administration Schemas
Query 1
Administrator
SQL Additions
Jump to
Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.
Context Help Snippets
Information
No connection established
Object Info Session
Output
Action Output
# Time Action Message Duration / Fetch
1 17:33:47 Could not connect, server may not be running. Unable to connect to localhost:3306


Manage Server Connections
MySQL Connections
Localhost
Connection Name: Localhost
Connection Remote Management System Profile
Information about the server and MySQL configuration, such as path to the configuration file, command to start or stop it etc. You may pick a preset configuration profile or customize one for your needs.
System Type: Windows
Installation Type: Windows (MySQL 5.5 Installer Package)
Configuration File: %ProgramData%\MySQL\MySQL Server 5.5\my.ini
Configuration File Section: mysqld
Windows Service Name: MySQL
Windows Hint:
When a connection is configured for "Windows Remote Management", the "Configuration File" must be specified using a mapped drive, network share or administrative share.
Examples of these are
M:\<path to file>\my.ini
\\<server>\<share>\<path to file>\my.ini
\\<server>\C$\Program Fiels\MySQL\MySQL Server 5.5\my.ini
New Delete Duplicate Move Up Move Down Test Connection Close

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*

Desarrollo:

| **Código del Programa** | Seudocódigo |
|---|---|
| `import tkinter as tk`<br>`from tkinter import messagebox, simpledialog, ttk`<br>`import json`<br>`import os`<br><br>`# ------------------------------`<br>`#   Datos iniciales`<br>`# ------------------------------`<br>`equipos = [`<br>`    {"id": 1, "nombre": "Laptop HP", "disponible": True, "precio": 15},`<br>`    {"id": 2, "nombre": "Laptop Dell", "disponible": True, "precio": 18},`<br>`    {"id": 3, "nombre": "PC Escritorio Lenovo", "disponible": True, "precio": 20},`<br>`    {"id": 4, "nombre": "PC Escritorio Acer", "disponible": True, "precio": 22},` | INICIO PROGRAMA "Sistema de Alquiler PinkTech"<br><br>---<br>INICIALIZAR LISTA equipos CON:<br>cada equipo tiene:<br>id, nombre, disponible (True), precio<br><br>INICIALIZAR LISTA alquileres VACÍA<br><br>---<br>FUNCIÓN guardar_datos():<br>ABRIR archivo "datos.json" en modo escritura |

```
    {"id": 5, "nombre": "Impresora Epson", "disponible":
True, "precio": 10},
    {"id": 6, "nombre": "Impresora Canon", "disponible":
True, "precio": 12},
    {"id": 7, "nombre": "Multifuncional HP", "disponible":
True, "precio": 14},
    {"id": 8, "nombre": "Laptop Asus", "disponible": True,
"precio": 16},
    {"id": 9, "nombre": "PC Escritorio Dell", "disponible":
True, "precio": 19},
    {"id": 10, "nombre": "Laptop Lenovo", "disponible":
True, "precio": 17},
    {"id": 11, "nombre": "Multifuncional Canon",
"disponible": True, "precio": 13}
]
alquileres = []
# ------------------------------
#   Funciones auxiliares
# ------------------------------
def guardar_datos():
    """Guarda datos en archivo JSON"""
    with open("datos.json", "w") as f:
        json.dump({"equipos": equipos, "alquileres":
alquileres}, f)
    messagebox.showinfo("🖫 Guardado", "Datos guardados
correctamente.")
def cargar_datos():
    """Carga datos si existe el archivo"""
    global equipos, alquileres
    if os.path.exists("datos.json"):
        with open("datos.json", "r") as f:
            datos = json.load(f)
            equipos[:] = datos.get("equipos", equipos)
            alquileres[:] = datos.get("alquileres",
alquileres)
def mostrar_equipos():
    disponibles = "\n".join(
        [f"{e['id']}. {e['nombre']} - ${e['precio']} -
{'Disponible ✓' if e['disponible'] else 'Ocupado ✗'}"
         for e in equipos]
    )
    messagebox.showinfo("✨ Equipos disponibles",
disponibles)
def buscar_equipo():
```

GUARDAR (equipos y alquileres) en formato JSON
MOSTRAR mensaje "Datos guardados correctamente"
FIN FUNCIÓN

---

FUNCIÓN cargar_datos():
SI archivo "datos.json" EXISTE EN EL DIRECTORIO:
ABRIR archivo
CARGAR datos JSON
ACTUALIZAR listas equipos y alquileres
FIN SI
FIN FUNCIÓN

---

FUNCIÓN mostrar_equipos():
CREAR texto con todos los equipos
PARA cada equipo EN equipos:
MOSTRAR id, nombre, precio y disponibilidad
MOSTRAR mensaje con la lista de equipos
FIN FUNCIÓN

---

FUNCIÓN buscar_equipo():
PEDIR al usuario un término de búsqueda
SI término está vacío:
SALIR de la función
FILTRAR equipos que contengan ese término en su nombre
SI hay resultados:
MOSTRAR lista de coincidencias
SINO:
MOSTRAR mensaje "No se encontró ningún equipo"
FIN FUNCIÓN

---

```
    termino = simpledialog.askstring("🔍 Buscar", "Ingrese
el nombre o tipo del equipo:")
    if not termino:
        return
    resultados = [e for e in equipos if termino.lower() in
e["nombre"].lower()]
    if resultados:
        texto = "\n".join(
            [f"{e['id']}. {e['nombre']} - ${e['precio']} -
{'Disponible ✅' if e['disponible'] else 'Ocupado ❌'}"
             for e in resultados]
        )
        messagebox.showinfo("Resultados de búsqueda", texto)
    else:
        messagebox.showinfo("Sin resultados", "No se
encontró ningún equipo con ese nombre.")


def alquilar_equipo():
    try:
        id_equipo = simpledialog.askinteger("💻 Alquiler",
"Ingrese el ID del equipo a alquilar:")
        if not id_equipo:
            return
        cliente = simpledialog.askstring("👤💼 Cliente",
"Ingrese el nombre del cliente:")
        if not cliente:
            messagebox.showwarning("Error", "Debe ingresar
el nombre del cliente.")
            return
        dias = simpledialog.askinteger("🧮 Días", "¿Cuántos
días desea alquilar el equipo?")
        if not dias or dias <= 0:
            messagebox.showwarning("Error", "Debe ingresar
un número válido de días.")
            return

        for e in equipos:
            if e["id"] == id_equipo:
                if e["disponible"]:
                    total = e["precio"] * dias
                    if dias > 5:
                        total *= 0.9
                    alquiler = {"cliente": cliente,
"equipo": e["nombre"], "dias": dias, "total": total}
```

FUNCIÓN alquilar_equipo():
PEDIR al usuario ID del equipo, nombre del cliente y días de alquiler
VALIDAR que los datos sean correctos

BUSCAR el equipo con el ID dado
SI equipo encontrado:
SI está disponible:
CALCULAR total = precio * días
SI días > 5:
APLICAR descuento del 10%
CREAR registro de alquiler con cliente, equipo, días y total
AÑADIR registro a la lista alquileres
MARCAR equipo como no disponible
MOSTRAR mensaje con la información del alquiler
PREGUNTAR si desea generar factura
SI respuesta es SÍ:
LLAMAR a generar_factura(alquiler)
LLAMAR a guardar_datos()
SINO:
MOSTRAR mensaje "Equipo no disponible"
SINO:
MOSTRAR mensaje "ID no encontrado"
FIN FUNCIÓN

---

FUNCIÓN devolver_equipo():
PEDIR nombre del cliente
BUSCAR en la lista de alquileres un registro con ese cliente
SI se encuentra:
MARCAR el equipo correspondiente como disponible
ELIMINAR el registro del alquiler
MOSTRAR mensaje "Devolución exitosa"

```
                    alquileres.append(alquiler)
                    e["disponible"] = False
                    messagebox.showinfo("🌸 Alquiler
confirmado",
                                        f"Equipo:
{e['nombre']}\nCliente: {cliente}\nDías: {dias}\nTotal:
${total:.2f}")
                    factura = messagebox.askyesno("🧾
Factura", "¿Desea generar factura?")
                    if factura:
                        generar_factura(alquiler)
                    guardar_datos()
                    return
                else:
                    messagebox.showwarning("Error", "Ese
equipo no está disponible.")
                    return
        messagebox.showerror("Error", "ID de equipo no
encontrado.")
    except Exception as ex:
        messagebox.showerror("Error", f"Ocurrió un error:
{ex}")


def devolver_equipo():
    cliente = simpledialog.askstring("🔁 Devolución",
"Ingrese el nombre del cliente:")
    if not cliente:
        return
    for a in alquileres:
        if a["cliente"].lower() == cliente.lower():
            for e in equipos:
                if e["nombre"] == a["equipo"]:
                    e["disponible"] = True
            alquileres.remove(a)
            messagebox.showinfo("💖 Devolución exitosa",
f"El equipo '{a['equipo']}' fue devuelto por {cliente}.")
            guardar_datos()
            return
    messagebox.showwarning("No encontrado", "No se encontró
ningún alquiler con ese nombre.")


def mostrar_alquileres():
    if not alquileres:
```

GUARDAR datos actualizados
SINO:
MOSTRAR mensaje "No se encontró ningún alquiler con ese nombre"
FIN FUNCIÓN

---

FUNCIÓN mostrar_alquileres():
SI la lista de alquileres está vacía:
MOSTRAR "No hay registros"
SINO:
MOSTRAR lista de todos los alquileres activos
FIN FUNCIÓN

---

FUNCIÓN generar_factura(alquiler):
CREAR texto con datos del cliente, equipo, días y total
MOSTRAR mensaje con la factura
GUARDAR factura en un archivo .txt con el nombre del cliente
FIN FUNCIÓN

---

INTERFAZ GRÁFICA (usando Tkinter):

CREAR ventana principal con título "Sistema de Alquiler - PinkTech"
CONFIGURAR tamaño 600x500, color de fondo rosa pastel

AGREGAR título decorativo en la parte superior

CREAR botones con estilo moderno y colores femeninos:
- Mostrar Equipos → llamar mostrar_equipos()
- Buscar Equipo → llamar buscar_equipo()

```
        messagebox.showinfo("📦 Alquileres", "No hay
registros de alquiler.")
    else:
        texto = "\n".join([f"{a['cliente']} - {a['equipo']}
({a['dias']} días) ${a['total']:.2f}" for a in alquileres])
        messagebox.showinfo("💼 Alquileres activos", texto)


def generar_factura(alquiler):
    factura_texto = (
        f"--- 💖 FACTURA ---\n"
        f"Cliente: {alquiler['cliente']}\n"
        f"Equipo: {alquiler['equipo']}\n"
        f"Días: {alquiler['dias']}\n"
        f"Total: ${alquiler['total']:.2f}\n"
        f"-----------------------"
    )
    messagebox.showinfo("Factura generada", factura_texto)
    with open(f"Factura_{alquiler['cliente']}.txt", "w") as
f:
        f.write(factura_texto)


# -------------------------------
#   Interfaz gráfica (femenina 💅)
# -------------------------------
ventana = tk.Tk()
ventana.title("🌷 Sistema de Alquiler - Empresa PinkTech")
ventana.geometry("600x500")
ventana.config(bg="#ffeef8")

# Colores suaves
color_fondo = "#ffeef8"
color_botones = "#f3a4b5"
color_hover = "#ffb6c1"
color_texto = "#ffffff"

# Título
titulo = tk.Label(
    ventana,
    text="🌸 Sistema de Alquiler de Equipos 🌸",
    font=("Comic Sans MS", 20, "bold"),
    bg=color_fondo,
    fg="#d63384"
)
```

- Alquilar Equipo → llamar alquilar_equipo()
- Devolver Equipo → llamar devolver_equipo()
- Ver Alquileres → llamar mostrar_alquileres()
- Guardar Datos → llamar guardar_datos()
- Salir → cerrar la ventana

AGREGAR texto decorativo al pie de página con el nombre de la creadora 💖

LLAMAR cargar_datos() para traer información previa

EJECUTAR ventana principal hasta que el usuario cierre la aplicación

FIN PROGRAMA

```
titulo.pack(pady=20)

# Marco principal
frame_botones = tk.Frame(ventana, bg=color_fondo)
frame_botones.pack(pady=10)

# Función para crear botones bonitos
def crear_boton(texto, comando):
    btn = tk.Button(
        frame_botones,
        text=texto,
        bg=color_botones,
        fg=color_texto,
        font=("Century Gothic", 12, "bold"),
        width=28,
        height=2,
        relief="flat",
        bd=0,
        cursor="hand2",
        command=comando,
        activebackground=color_hover,
        activeforeground="white"
    )
    btn.pack(pady=6)
    btn.bind("<Enter>", lambda e:
btn.config(bg=color_hover))
    btn.bind("<Leave>", lambda e:
btn.config(bg=color_botones))
    return btn

# Botones principales
crear_boton("💻 Mostrar Equipos", mostrar_equipos)
crear_boton("🔍 Buscar Equipo", buscar_equipo)
crear_boton("🎁 Alquilar Equipo", alquilar_equipo)
crear_boton("🔁 Devolver Equipo", devolver_equipo)
crear_boton("📋 Ver Alquileres", mostrar_alquileres)
crear_boton("💾 Guardar Datos", guardar_datos)
crear_boton("🚪 Salir del Sistema", ventana.destroy)

# Pie decorativo
footer = tk.Label(
    ventana,
    text="Hecho con 💖 por Eliana | PinkTech 2025",
    bg=color_fondo,
    fg="#a84e6c",
```

```
    font=("Lucida Handwriting", 10)
)
footer.pack(side="bottom", pady=15)

# Cargar datos previos
cargar_datos()

ventana.mainloop()
```


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.