Universidad Tecnológica de Panamá
Facultad de Sistemas
Computacionales Asignatura:
Desarrollo Lógico y Algoritmo Taller
Práctico1

Profesor: Napoleón Ibarra Valor: 100 puntos

Estudiante: Jacob Samudio

Diógenes Serrano

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

Desarrollo

Primera Parte

1.1 Búsqueda Secuencial
Concepto:
La búsqueda secuencial (o búsqueda lineal) es un algoritmo simple que recorre cada elemento de un arreglo uno por uno, desde el inicio hasta el final, comparando cada elemento con el valor buscado. Es el método más básico de búsqueda y no requiere que el arreglo esté ordenado.
Características:

- Complejidad temporal: O(n) en el peor caso
- No requiere arreglo ordenado
- Fácil de implementar
- Eficiente para arreglos pequeños

Sintaxis General:

```
for elemento in arreglo:
    if elemento == valor_buscado:
        return posicion
```

Ejemplo Funcional en Python:

```
def busqueda_secuencial(arreglo, objetivo):
    """
    Busca un elemento en un arreglo de forma secuencial
    Retorna la posición si lo encuentra, -1 si no
    """
    for indice in range(len(arreglo)):
        if arreglo[indice] == objetivo:
            return indice
```

```
    return -1

# Ejemplo de uso con equipos de computación
equipos = ["Laptop HP", "PC Dell", "Impresora Epson", "Laptop Lenovo", "Tablet Samsung"]

# Buscar un equipo específico
equipo_buscado = "PC Dell"
resultado = busqueda_secuencial(equipos, equipo_buscado)

if resultado != -1:
    print(f" Equipo '{equipo_buscado}' encontrado en la posición {resultado}")
else:
    print(f" Equipo '{equipo_buscado}' no encontrado en el inventario")
```

Simulación del Proceso:

```
Arreglo: ["Laptop HP", "PC Dell", "Impresora Epson", "Laptop Lenovo", "Tablet Samsung"]
Buscando: "PC Dell"

Paso 1: Comparar "Laptop HP" == "PC Dell" → No
Paso 2: Comparar "PC Dell" == "PC Dell" → Sí ✓
Resultado: Encontrado en posición 1
```

1.2 Push Down

Concepto:

El algoritmo Push Down (o "empujar hacia abajo") mueve un elemento específico a la primera posición del arreglo, desplazando todos los elementos anteriores una posición hacia la derecha. Es útil para reordenar elementos basándose en frecuencia de uso o prioridad.

Características:

- Reordena elementos sin cambiar el tamaño del arreglo
- Mantiene todos los elementos originales
- Útil para listas de acceso frecuente

Sintaxis General:

```
posicion = arreglo.index(elemento)
if posicion > 0:
    elemento = arreglo.pop(posicion)
    arreglo.insert(0, elemento)
```

Ejemplo Funcional en Python:

```
def aplicar_push_down(arreglo, elemento_prioritario):
    """
    Mueve un elemento a la primera posición usando Push Down
    """
    try:
        # Encontrar la posición del elemento
        posicion_actual = arreglo.index(elemento_prioritario)

        # Solo aplicar si no está ya en primera posición
        if posicion_actual > 0:
            # Remover el elemento de su posición actual
            elemento_movido = arreglo.pop(posicion_actual)
            # Insertar en la primera posición
            arreglo.insert(0, elemento_movido)
            return True
        return False
    except ValueError:
        print(f"Elemento '{elemento_prioritario}' no encontrado en el arreglo")
        return False

# Ejemplo con clientes frecuentes
```

```python
clientes = ["Ana López", "Carlos Ruiz", "María García", "Juan Pérez", "Laura Martínez"]

print(" Lista original de clientes:")
for i, cliente in enumerate(clientes):
    print(f"  {i+1}. {cliente}")

# Aplicar Push Down para priorizar un cliente
cliente_prioritario = "Juan Pérez"
aplicar_push_down(clientes, cliente_prioritario)

print(f"\n Después de Push Down (priorizando a '{cliente_prioritario}'):")
for i, cliente in enumerate(clientes):
    print(f"  {i+1}. {cliente}")
```

Simulación del Proceso:

```
Arreglo original: ["Ana López", "Carlos Ruiz", "María García", "Juan Pérez", "Laura Mar
tínez"]
Elemento a mover: "Juan Pérez" (posición 3)

Proceso:
1. Remover "Juan Pérez" de la posición 3
2. Desplazar elementos: ["Ana López", "Carlos Ruiz", "María García", "Laura Martínez"]
3. Insertar "Juan Pérez" en posición 0

Resultado: ["Juan Pérez", "Ana López", "Carlos Ruiz", "María García", "Laura Martínez"]
```

2. Base de Datos MySQL
Definición:
MySQL es un sistema de gestión de bases de datos relacional (RDBMS) de código abierto desarrollado por Oracle Corporation. Utiliza el lenguaje SQL (Structured Query Language) para gestionar, manipular y consultar datos almacenados en tablas relacionadas.
Requisitos para Instalación:
Requisitos Mínimos del Sistema:

- **Sistema Operativo:** Windows 7+, Linux (Ubuntu, CentOS), macOS 10.10+
- **Procesador:** 1 GHz o superior
- **Memoria RAM:** Mínimo 2 GB (recomendado 4 GB)
- **Espacio en Disco:** 500 MB para MySQL + espacio para datos
- **Python:** Versión 3.6 o superior

Componentes Necesarios:

1. **MySQL Server:** Motor principal de base de datos
2. **MySQL Workbench:** Herramienta gráfica de administración (opcional)
3. **MySQL Connector/Python:** Conector oficial para Python
4. **Editor de Código:** VS Code, PyCharm, o cualquier editor de texto

Conexión Python-MySQL:
Instalación del Conector:

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

Código de Conexión Básica:

```python
import mysql.connector
from mysql.connector import Error

def conectar_mysql():
    try:
        # Establecer conexión con la base de datos
        conexion = mysql.connector.connect(
            host='localhost',      # Servidor donde está MySQL
            user='root',           # Usuario de la base de datos
            password='',           # Contraseña (vacía por defecto en XAMPP)
```

```
            database='empresa_xyz' # Nombre de la base de datos
        )

        # Verificar si la conexión fue exitosa
        if conexion.is_connected():
            print(" Conexión exitosa a MySQL")
            print(f" Versión del servidor: {conexion.get_server_info()}")
            return conexion

    except Error as e:
        print(f" Error al conectar a MySQL: {e}")
        return None

# Ejemplo de uso
conexion = conectar_mysql()
if conexion:
    # Realizar operaciones con la base de datos...
    conexion.close()
```

Operaciones Básicas CRUD:

```
# INSERTAR datos
def insertar_equipo(conexion, nombre, tipo, precio):
    cursor = conexion.cursor()
    query = "INSERT INTO equipos (nombre, tipo, precio_hora) VALUES (%s, %s, %s)"
    valores = (nombre, tipo, precio)
    cursor.execute(query, valores)
    conexion.commit()
    print("Equipo insertado correctamente")

# CONSULTAR datos
def consultar_equipos(conexion):
    cursor = conexion.cursor()
    cursor.execute("SELECT * FROM equipos")
    resultados = cursor.fetchall()
    for equipo in resultados:
        print(equipo)
```

¿Qué es una Replicación en Base de Datos?
Definición:
La replicación de bases de datos es el proceso de copiar y mantener múltiples copias idénticas de una base de datos en diferentes servidores. Esto permite distribuir la carga de trabajo y mejorar la disponibilidad de los datos.
Tipos de Replicación:

1. **Replicación Maestro-Esclavo:**
    - Un servidor principal (maestro) maneja todas las escrituras
    - Múltiples servidores secundarios (esclavos) manejan lecturas
    - Los esclavos se sincronizan con el maestro
2. **Replicación Maestro-Maestro:**
    - Múltiples servidores pueden leer y escribir
    - Los cambios se sincronizan entre todos los servidores
    - Mayor complejidad de configuración
3. **Replicación Circular:**
    - Cada servidor replica al siguiente en una cadena
    - Útil para distribuir carga geográficamente

Ventajas de la Replicación:

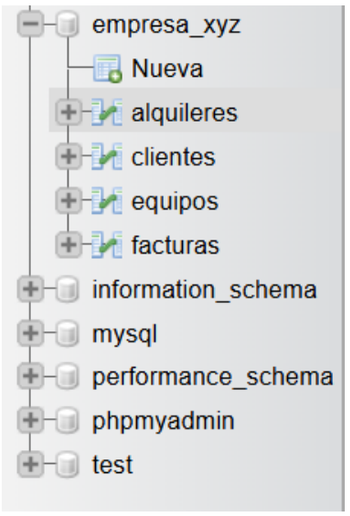
- Alta Disponibilidad: Si un servidor falla, otros pueden tomar el control
- Mejor Rendimiento: Distribuye la carga de consultas
- Respaldos en Tiempo Real: Datos replicados inmediatamente
- Escalabilidad: Fácil agregar más servidores según crece la demanda

Ejemplo de Escenario de Replicación:

```
Servidor Principal (Maestro) en Panamá
        ↓
Servidor Réplica (Esclavo) en Costa Rica ↔ Servidor Réplica (Esclavo) en Colombia

Todos los servidores contienen la misma base de datos 'empresa_xyz'
Las escrituras van al maestro, las lecturas se distribuyen entre esclavos
```

Segunda parte:



Objetivo: Configurar el entorno de base de datos

- **Instalar y configurar MySQL** en el equipo local
- **Crear una base de datos** con nombre a elección (empresa_xyz)
- **Verificar funcionamiento** del gestor de base de datos
- **Preparar la conexión** para el prototipo Python-MySQL

	id	id_equipo	cliente	horas	costo_total	fecha_inicio	fecha_fin	estado
Editar Copiar Borrar	11	1	Juan Pablo	3	7.50	2025-11-10 18:01:23	2025-11-10 18:03:46	Finalizado
Editar Copiar Borrar	12	5	Juan Pablo	7	10.50	2025-11-10 18:02:04	2025-11-11 15:27:44	Finalizado

	id	nombre	tipo	estado	precio_hora
Editar Copiar Borrar	1	Laptop HP EliteBook	Laptop	Disponible	2.50
Editar Copiar Borrar	2	Laptop Dell Latitude	Laptop	Disponible	2.75
Editar Copiar Borrar	3	PC Escritorio Lenovo	PC Escritorio	Disponible	2.00
Editar Copiar Borrar	4	PC Escritorio HP	PC Escritorio	Disponible	2.25
Editar Copiar Borrar	5	Impresora Epson	Impresora Multifuncional	Disponible	1.50
Editar Copiar Borrar	6	Laptop Asus VivoBook	Laptop	Disponible	2.25
Editar Copiar Borrar	7	PC Escritorio Dell	PC Escritorio	Disponible	2.00
Editar Copiar Borrar	8	Impresora Canon	Impresora Multifuncional	Disponible	1.75
Editar Copiar Borrar	9	Laptop Lenovo ThinkPad	Laptop	Disponible	3.00
Editar Copiar Borrar	10	PC Escritorio Acer	PC Escritorio	Disponible	1.75
Editar Copiar Borrar	11	Impresora Brother	Impresora Multifuncional	Mantenimiento	1.50

Tercera parte:

Caso de Estudio: Empresa XYZ con servicios de alquiler de equipos

- **Programar en Python** un sistema de alquiler de equipos
- **Gestionar 11 equipos:** 4 laptops, 4 PCs escritorio, 3 impresoras multifuncionales
- **Funcionalidades requeridas:**
  - Insertar, actualizar y eliminar registros
  - Asignar/alquilar equipos a clientes
  - Calcular costos finales basados en tiempo de uso
  - Generar facturas cuando el cliente las solicite
  - Almacenar todos los registros permanentemente
  - Controlar estado de equipos (activos/no activos)

Entregable: Código Python funcional con pseudocódigo


```
Algoritmo AlquilerEquipos

    // 11 equipos: 4 laptops, 4 PCs, 3 impresoras
    Dimension equipos[11,3]  // nombre, estado, precio_hora

    // Inicializar equipos
    equipos[1,1] = "Laptop1"; equipos[1,2] = "Disponible"; equipos[1,3] = "2.5"
    equipos[2,1] = "Laptop2"; equipos[2,2] = "Disponible"; equipos[2,3] = "2.5"
    equipos[3,1] = "Laptop3"; equipos[3,2] = "Disponible"; equipos[3,3] = "2.5"
    equipos[4,1] = "Laptop4"; equipos[4,2] = "Disponible"; equipos[4,3] = "2.0"
    equipos[5,1] = "PC1"; equipos[5,2] = "Disponible"; equipos[5,3] = "1.5"
    equipos[6,1] = "PC2"; equipos[6,2] = "Disponible"; equipos[6,3] = "1.5"
    equipos[7,1] = "PC3"; equipos[7,2] = "Disponible"; equipos[7,3] = "1.5"
    equipos[8,1] = "PC4"; equipos[8,2] = "Disponible"; equipos[8,3] = "1.25"
    equipos[9,1] = "Impresora1"; equipos[9,2] = "Disponible"; equipos[9,3] = "0.5"
    equipos[10,1] = "Impresora2"; equipos[10,2] = "Disponible"; equipos[10,3] = "0.75"
    equipos[11,1] = "Impresora3"; equipos[11,2] = "Disponible"; equipos[11,3] = "0.6"

    Repetir
        Escribir "1. Ver equipos"
        Escribir "2. Alquilar"
        Escribir "3. Devolver"
        Escribir "4. Salir"
        Leer opcion

        Segun opcion Hacer
            Caso 1:
                Para i = 1 Hasta 11 Hacer
                    Escribir i, ". ", equipos[i,1], " - ", equipos[i,2], " - $", equipos[i,3], "/hora"
                FinPara

            Caso 2:
                Escribir "Número del equipo: "
                Leer num
                Si equipos[num,2] = "Disponible" Entonces
                    Escribir "Horas: "
                    Leer horas
                    costo = ConvertirANumero(equipos[num,3]) * horas
                    equipos[num,2] = "Ocupado"
                    Escribir "Costo: $", costo
                Sino
                    Escribir "No disponible"
                FinSi

            Caso 3:
                Escribir "Número del equipo: "
                Leer num
                equipos[num,2] = "Disponible"
                Escribir "Equipo devuelto"

        FinSegun
        Escribir ""
    Hasta Que opcion = 4

FinAlgoritmo
```

Cuarta parte:

Objetivo: Diseñar la infraestructura de red para la empresa

- **Utilizar Cisco Packet Tracer** para crear el diagrama
- **Diseñar topología LAN** según el caso de estudio
- **Configurar todos los dispositivos:**
  - Router con funcionalidades básicas
  - Switches para conectividad
  - Servidor para base de datos
  - 11 equipos finales (PCs, laptops, impresoras)
  - Access Point para conectividad WiFi
- **Verificar conectividad** entre todos los dispositivos
- **Asignar IPs estáticas** a equipos críticos