

Patrón OBJECT POOL

Alexa Alpízar Mora C20281

Indice

1

- DEFINICIÓN

2

- PROBLEMA QUE
RESUELVE

3

- VENTAJAS Y
DESVENTAJAS

4

- CASOS COMUNES

5

- EJEMPLO

6

- FINAL



Definición

- Conjunto de objetos ya inicializados (creacional).

Soluciones

1

Tiempo de
 inicialización
(conexiones de base
 de datos, conexiones
 de red).

2

Consumo de
 recursos (hilos,
 procesos, objetos
 pesados).

3

Limitaciones del
 sistema (número
 máximo de
 conexiones
 permitidas).

Ventajas

- Mejora significativa del rendimiento al evitar creación repetida de objetos costosos.
- Control preciso sobre el número máximo de instancias.
- Proporciona estadísticas de uso del pool.

Desventajas

- Mayor complejidad en la implementación.
- Puede causar problemas de memoria si no se liberan objetos correctamente.
- Puede ocultar problemas de diseño si se usa innecesariamente.

Casos

1

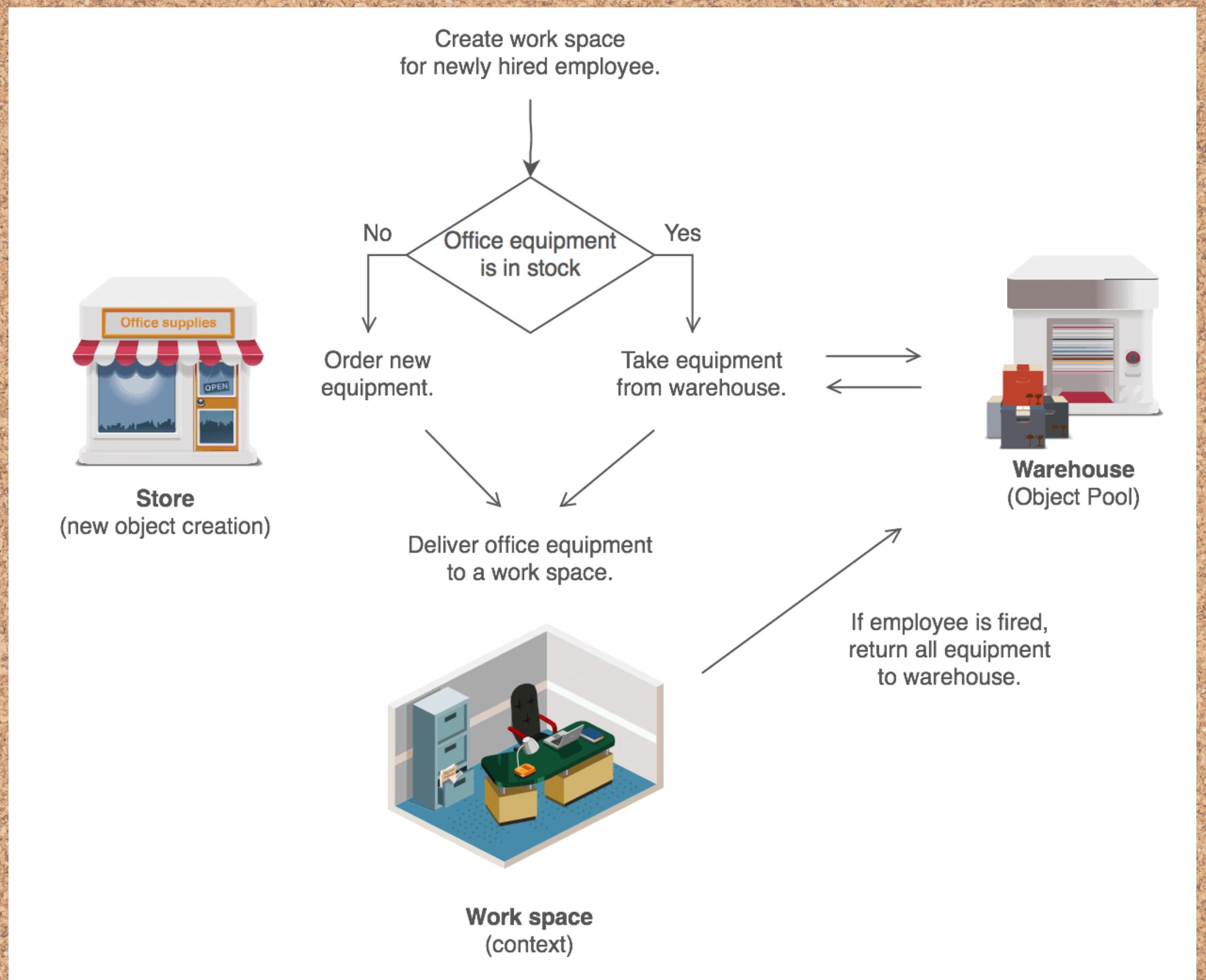
Conexiones de
base de datos:
reutilizar
conexiones en
lugar de crear
nuevas.

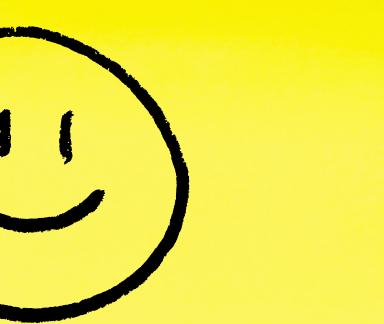
2

Hilos
(Thread Pool):
gestionar un
conjunto de hilos
reutilizables.

3

Conexiones de red:
para servidores
que manejan
múltiples clientes.





```
# connection_pool.py
import threading
import time
from typing import List, Optional

class DatabaseConnection:
    def __init__(self, connection_id: int):
        self.connection_id = connection_id
        self.in_use = False
        self.created_at = time.time()

    def connect(self):
        print(f"Conexión {self.connection_id} establecida")

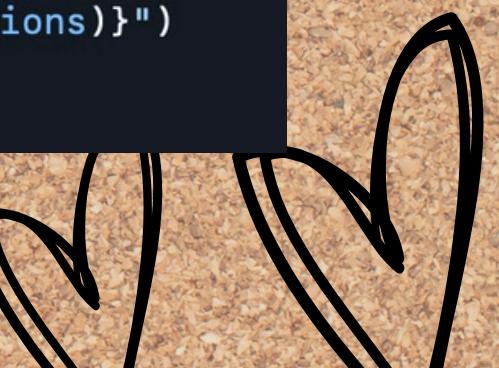
    def execute_query(self, query: str):
        print(f"Conexión {self.connection_id} ejecutando: {query}")

    def reset(self):
        """Resetea la conexión para reutilización"""
        self.in_use = False

class ConnectionPool:
    _instance = None
    _lock = threading.Lock()

    def __new__(cls):
        with cls._lock:
            if cls._instance is None:
                cls._instance = super().__new__(cls)
        return cls._instance

    def __init__(self):
        self._available_connections: List[DatabaseConnection] = []
        self._in_use_connections: List[DatabaseConnection] = []
        self.max_size = 5
        self._initialize_pool()
```

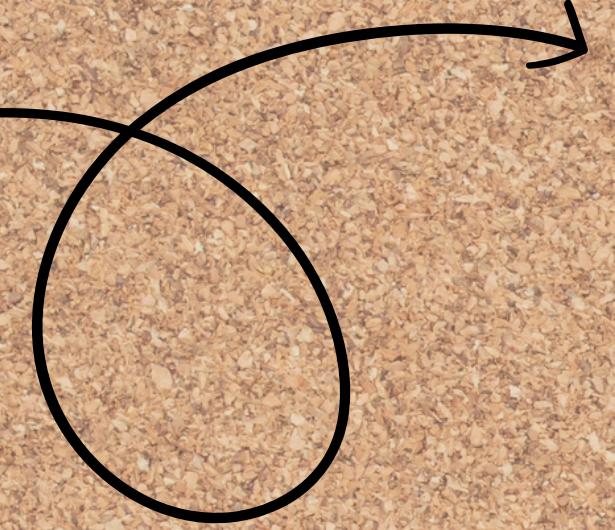


```
def _initialize_pool(self):
    """Inicializa el pool con conexiones listas para usar"""
    for i in range(self.max_size):
        connection = DatabaseConnection(i + 1)
        connection.connect()
        self._available_connections.append(connection)

def acquire_connection(self) -> Optional[DatabaseConnection]:
    """Obtiene una conexión del pool"""
    if self._available_connections:
        connection = self._available_connections.pop()
        connection.in_use = True
        self._in_use_connections.append(connection)
        print(f"Conexión {connection.connection_id} adquirida")
        return connection
    else:
        print("No hay conexiones disponibles en el pool")
        return None

def release_connection(self, connection: DatabaseConnection):
    """Devuelve una conexión al pool"""
    if connection in self._in_use_connections:
        self._in_use_connections.remove(connection)
        connection.reset()
        self._available_connections.append(connection)
        print(f"Conexión {connection.connection_id} liberada")

def get_pool_status(self):
    """Muestra el estado actual del pool"""
    print("\nEstado del Pool:")
    print(f"Disponibles: {len(self._available_connections)}")
    print(f"En uso: {len(self._in_use_connections)}")
    print(f"Total: {self.max_size}")
```



```
# Uso del Connection Pool
def main():
    # Crear el pool (singleton)
    pool = ConnectionPool()

    # Adquirir y usar conexiones
    conn1 = pool.acquire_connection()
    conn2 = pool.acquire_connection()
    conn3 = pool.acquire_connection()

    if conn1:
        conn1.execute_query("SELECT * FROM users")
    if conn2:
        conn2.execute_query("INSERT INTO logs VALUES ('test')")

    pool.get_pool_status()

    # Liberar conexiones
    if conn1:
        pool.release_connection(conn1)
    if conn2:
        pool.release_connection(conn2)

    pool.get_pool_status()

    # Adquirir más conexiones (deberían reutilizarse)
    conn4 = pool.acquire_connection()
    conn5 = pool.acquire_connection()

    pool.get_pool_status()

if __name__ == "__main__":
    main()
```





¡Gracias!

