Solucionarios

Semana 5

20.07.2025 - 27.07.2025

Caso General

Gestión de una Biblioteca Digital

Las bibliotecas, tanto físicas como digitales, necesitan un sistema robusto para catalogar sus libros, gestionar los préstamos y devoluciones, y saber qué libros están disponibles. La automatización de estos procesos es esencial para un servicio eficiente.

- Yamil Ayma

Nivel Fácil: Buscar un Libro por Título

```python
def buscar_libro_por_titulo(catalogo, titulo_buscado):
    """
    Busca un libro en el catálogo por su título y devuelve el diccionario del libro.
    """
    # 1. Recorremos cada libro (que es un diccionario) en la lista del catálogo.
    for libro in catalogo:
        # 2. Comparamos el título del libro actual con el título que buscamos.
        if libro['titulo'] == titulo_buscado:
            # 3. Si coinciden, hemos encontrado el libro. Lo devolvemos y la función termina.
            return libro

    # 4. Si el bucle termina y no hemos encontrado el libro, significa que no está.
    #    Devolvemos None.
    return None

# --- Pruebas ---
def probar_buscar_libro_por_titulo():
    catalogo = [
        {'titulo': 'Cien Años de Soledad', 'autor': 'Gabriel García Márquez', 'disponible': True},
        {'titulo': 'El Señor de los Anillos', 'autor': 'J.R.R. Tolkien', 'disponible': False},
        {'titulo': '1984', 'autor': 'George Orwell', 'disponible': True}
    ]

    libro_encontrado = buscar_libro_por_titulo(catalogo, '1984')
    esperado1 = {'titulo': '1984', 'autor': 'George Orwell', 'disponible': True}
    print(f"Prueba 1: {libro_encontrado == esperado1}")

    libro_no_encontrado = buscar_libro_por_titulo(catalogo, 'Fahrenheit 451')
    print(f"Prueba 2: {libro_no_encontrado is None}")

    libro_en_catalogo_vacio = buscar_libro_por_titulo([], 'Cien Años de Soledad')
    print(f"Prueba 3: {libro_en_catalogo_vacio is None}")

probar_buscar_libro_por_titulo()
```

★ Click para acceder al código ★

Nivel Medio: Prestar y Devolver un Libro

```python
from facil import buscar_libro_por_titulo

def gestionar_prestamo(catalogo, titulo, accion):
    """
    Gestiona el préstamo o devolución de un libro, actualizando su estado.
    """
    # 1. Recorremos el catálogo para encontrar el libro por su título.
    for libro in catalogo:
        if libro['titulo'] == titulo:
            # --- Libro encontrado. Ahora procesamos la acción ---

            # 2. Si la acción es "prestar"...
            if accion == "prestar":
                # ...verificamos si está disponible.
                if libro['disponible']:
                    # Si lo está, lo marcamos como no disponible y devolvemos True.
                    libro['disponible'] = False
                    return True
                else:
                    # Si no está disponible, no se puede prestar. Devolvemos False.
                    return False

            # 3. Si la acción es "devolver"...
            elif accion == "devolver":
                # ...verificamos si NO está disponible (es decir, está prestado).
                if not libro['disponible']:
                    # Si está prestado, lo marcamos como disponible y devolvemos True.
                    libro['disponible'] = True
                    return True
                else:
                    # Si ya está disponible, no se puede devolver. Devolvemos False.
                    return False

    # 4. Si el bucle termina, el libro no se encontró en el catálogo. Devolvemos False.
    return False

# --- Pruebas ---
# (Se necesita la función buscar_libro_por_titulo para las pruebas)
def probar_gestionar_prestamo():
    catalogo_base = [
        {'titulo': 'Cien Años de Soledad', 'autor': 'G.G. Márquez', 'disponible': True},
        {'titulo': 'El Señor de los Anillos', 'autor': 'J.R.R. Tolkien', 'disponible': False}
    ]

    catalogo_p1 = [libro.copy() for libro in catalogo_base]
    resultado1 = gestionar_prestamo(catalogo_p1, 'Cien Años de Soledad', 'prestar')
    libro_actualizado1 = buscar_libro_por_titulo(catalogo_p1, 'Cien Años de Soledad')
    print(f"Prueba 1 (Prestar éxito): {resultado1 is True and not libro_actualizado1['disponible']}")

    catalogo_p2 = [libro.copy() for libro in catalogo_base]
    resultado2 = gestionar_prestamo(catalogo_p2, 'El Señor de los Anillos', 'prestar')
    print(f"Prueba 2 (Prestar fallo): {resultado2 is False}")

    catalogo_p3 = [libro.copy() for libro in catalogo_base]
    resultado3 = gestionar_prestamo(catalogo_p3, 'El Señor de los Anillos', 'devolver')
    libro_actualizado3 = buscar_libro_por_titulo(catalogo_p3, 'El Señor de los Anillos')
    print(f"Prueba 3 (Devolver éxito): {resultado3 is True and libro_actualizado3['disponible']}")

probar_gestionar_prestamo()
```

★ Click para acceder al código ★

Nivel Avanzado: Reporte de Libros Vencidos

```python
from datetime import date, timedelta

def reporte_libros_vencidos(catalogo, fecha_actual):
    """
    Genera un reporte de libros con fecha de devolución vencida,
    ordenados por días de retraso.
    """
    # 1. Creamos una lista vacía para guardar los libros vencidos.
    libros_con_retraso = []

    # 2. Recorremos el catálogo para revisar cada libro.
    for libro in catalogo:
        # 3. Verificamos tres cosas:
        #    a) que el libro NO esté disponible (está prestado).
        #    b) que el libro TENGA una fecha de devolución.
        #    c) que esa fecha de devolución SEA ANTERIOR a la fecha actual.
        if (not libro['disponible'] and
                'fecha_devolucion' in libro and
                libro['fecha_devolucion'] < fecha_actual):

            # 4. Si se cumplen las condiciones, calculamos los días de retraso.
            dias_retraso = (fecha_actual - libro['fecha_devolucion']).days

            # 5. Creamos un diccionario para el reporte y lo añadimos a nuestra lista.
            libros_con_retraso.append({
                'titulo': libro['titulo'],
                'dias_retraso': dias_retraso
            })

    # 6. Ordenamos la lista de libros con retraso.
    #    key=lambda item: item['dias_retraso'] le dice a sorted que ordene usando el número de días.
    #    reverse=True ordena de mayor a menor.
    reporte_ordenado = sorted(libros_con_retraso, key=lambda item: item['dias_retraso'], reverse=True)

    # 7. Devolvemos la lista ordenada.
    return reporte_ordenado

# --- Pruebas ---
def probar_reporte_libros_vencidos():
    hoy = date(2023, 10, 27)
    catalogo = [
        {'titulo': '1984', 'autor': 'George Orwell', 'disponible': False, 'fecha_devolucion':
date(2023, 10, 20)},
        {'titulo': 'Dune', 'autor': 'Frank Herbert', 'disponible': False, 'fecha_devolucion':
date(2023, 10, 30)},
        {'titulo': 'El Hobbit', 'autor': 'J.R.R. Tolkien', 'disponible': True},
        {'titulo': 'Fahrenheit 451', 'autor': 'Ray Bradbury', 'disponible': False, 'fecha_devolucion':
date(2023, 9, 27)}
    ]

    reporte = reporte_libros_vencidos(catalogo, hoy)
    esperado1 = [
        {'titulo': 'Fahrenheit 451', 'dias_retraso': 30},
        {'titulo': '1984', 'dias_retraso': 7}
    ]
    print(f"Prueba 1: {reporte == esperado1}")

    catalogo_sin_vencidos = [
        {'titulo': 'Dune', 'autor': 'Frank Herbert', 'disponible': False, 'fecha_devolucion':
date(2023, 11, 1)}
    ]
    reporte2 = reporte_libros_vencidos(catalogo_sin_vencidos, hoy)
    print(f"Prueba 2: {reporte2 == []}")

probar_reporte_libros_vencidos()
```

★ Click para acceder al código ★

Éxitos en los ejercicios

Hay dos maneras de construir un diseño de software. Una es simplificarlo tanto que no presente deficiencias evidentes. Y la otra es complicarlo tanto que no presente deficiencias evidentes.

- CAR Hoare

Aprende mucho