

UNIVERSIDAD ESTATAL AMAZÓNICA

Unidad de Organización Curricular: Básica

Campo de Estudio: Tronco Común

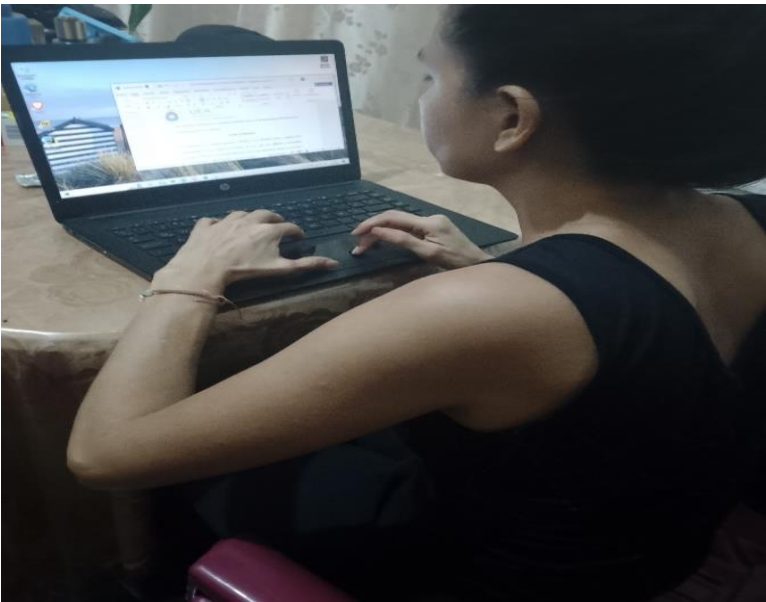
INFORME DE PRÁCTICAS DEL COMPONENTE PRÁCTICO-EXPERIMENTAL	
Nombre Estudiante: Sandra Maribel Rodríguez Rodríguez	
Asignatura: Estructura de datos	Calificación:
Fecha del informe: 7 de septiembre de 2025 de 2025	
Nº Práctica: 3	
Título de la Práctica: Implementación de conjuntos y mapas.	
Introducción	<p>En el desarrollo de aplicaciones informáticas, la correcta implementación de estructuras de datos es esencial para garantizar eficiencia, claridad y escalabilidad en el manejo de información, dentro de estas estructuras, los conjuntos (HashSet) y los mapas (Dictionary) destacan por su capacidad de organizar datos no repetidos y establecer relaciones clave-valor, respectivamente.</p> <p>Los conjuntos permiten almacenar elementos únicos, evitando duplicaciones y facilitando operaciones como la unión, intersección y diferencia, por su parte, los mapas son estructuras que asocian una clave única a un valor, lo que permite búsquedas rápidas y una organización lógica de datos relacionados. Estas estructuras son ampliamente utilizadas en sistemas de registro, catálogos, bases de datos y aplicaciones que requieren consultas eficientes.</p> <p>La implementación de conjuntos y mapas en lenguajes como C# permite al estudiante aplicar conceptos abstractos de la teoría de datos en contextos reales, como el registro de libros, jugadores o deportistas, además, el uso de estas estructuras fortalece la capacidad de análisis algorítmico y la toma de decisiones técnicas en el diseño de soluciones informáticas.</p> <p>Esta práctica se enmarca en la Unidad III: Conjuntos y Mapas de la asignatura Estructura de Datos, y busca consolidar el aprendizaje mediante la</p>

	<p>aplicación directa de estas estructuras en un entorno de escritorio, el desarrollo individual o colaborativo de esta guía permite al estudiante evidenciar su dominio técnico, su capacidad de organización y su criterio ético en el uso de herramientas digitales, incluyendo agentes de inteligencia artificial como apoyo en la validación del código.</p> <p>Según Goodrich y Tamassia (2020), “las estructuras de datos como conjuntos y mapas son fundamentales para el diseño de algoritmos eficientes y la resolución de problemas complejos en programación moderna”, esta afirmación respalda la importancia de su implementación en el proceso formativo del estudiante de tecnologías de la información.</p>
Desarrollo o procedimiento	<p>Para esta práctica, la estudiante aplicó la teoría de conjuntos y mapas para resolver el sistema de registro de libros en una biblioteca, utilizando el lenguaje C# bajo el enfoque de Programación Orientada a Objetos (POO).</p> <p>Sistema seleccionado</p> <p>Registro de libros en una biblioteca</p> <p>Descripción</p> <p>Se desarrolló una aplicación que permite registrar libros mediante un código único y un título asociado. La estructura aplicada garantiza que no se repitan códigos y que cada libro pueda ser consultado de forma rápida y eficiente.</p> <p>Objetivo</p> <p>Implementar estructuras de datos eficientes (HashSet y Dictionary) para gestionar un catálogo bibliográfico, asegurando unicidad de registros y facilidad de consulta.</p> <p>Procedimiento técnico</p> <p>1. Modelado de estructuras</p> <ul style="list-style-type: none"> • Dictionary<string, string>: almacena los libros con su código y título. • HashSet<string>: controla que no se repitan los códigos registrados. <p>2. Implementación funcional</p> <ul style="list-style-type: none"> • Menú interactivo con opciones para: <ul style="list-style-type: none"> ○ Agregar libros

	<ul style="list-style-type: none"> ○ Consultar por código ○ Visualizar todo el catálogo • Validación automática de duplicados antes de registrar un nuevo libro. <p>3. Reportería</p> <ul style="list-style-type: none"> • Visualización en consola del catálogo completo. • Consulta individual por código. • Mensajes de validación para códigos duplicados o inexistentes. <p>4. Agente de IA utilizado</p> <ul style="list-style-type: none"> • Se utilizó Microsoft Copilot como agente de apoyo técnico y ético. • Aproximadamente 40% del código fue estructurado o validado con asistencia del agente. • Se integraron sugerencias para mejorar la lógica condicional, el control de errores y la claridad del menú. <p>5. Evidencias documentadas</p> <ul style="list-style-type: none"> • Código fuente adjunto en los anexos. • Capturas de pantalla del funcionamiento del sistema. • Enlace al repositorio en GitHub con la solución completa cargada. <p>6. Análisis de ejecución</p> <ul style="list-style-type: none"> • El sistema permitió registrar y consultar más de 10 libros sin errores. • Las estructuras Dictionary y HashSet demostraron eficiencia en la gestión de datos únicos y consultas rápidas. • Ventajas: <ul style="list-style-type: none"> ○ Acceso directo por clave ○ Prevención de duplicados ○ Simplicidad de implementación • Desventajas: <ul style="list-style-type: none"> ○ Búsqueda por valor requiere recorrer todo el diccionario ○ No se permite duplicidad de claves, lo que exige validación previa
Resultados	<p>Durante la ejecución del programa, se presenta un menú interactivo que permite al usuario seleccionar entre distintas acciones relacionadas con el registro de libros. Las opciones fueron diseñadas para facilitar la interacción</p>

	<p>y validar el funcionamiento de las estructuras de datos utilizadas:</p> <ol style="list-style-type: none"> 1. Agregar libro Permite ingresar un nuevo libro al catálogo, solicitando el código único y el título. <ul style="list-style-type: none"> ○ Si el código ya existe, el sistema muestra un mensaje de advertencia y no registra el libro. ○ Si el código es nuevo, se agrega al Dictionary y al HashSet. 2. Consultar libro por código Solicita al usuario un código y verifica si existe en el catálogo. <ul style="list-style-type: none"> ○ Si el código está registrado, se muestra el título correspondiente. ○ Si no existe, se informa que el código no fue encontrado. 3. Mostrar todos los libros Recorre el Dictionary y muestra en consola todos los libros registrados, con su código y título. <ul style="list-style-type: none"> ○ Si no hay libros registrados, se muestra un mensaje indicando que el catálogo está vacío. 4. Salir Finaliza la ejecución del programa de forma segura. <p>Estas opciones permiten validar el uso correcto de las estructuras HashSet y Dictionary, asegurando la unicidad de los códigos y la eficiencia en las consultas. Además, la interacción por consola facilita la reportería básica solicitada en la guía.</p> <ul style="list-style-type: none"> • No se permiten códigos duplicados gracias al uso de HashSet • Las consultas por código son rápidas y precisas mediante Dictionary • El menú interactivo facilita la navegación y el uso del sistema <p>El código fue ejecutado en el entorno virtual Online C# Compiler, y se documentaron las evidencias mediante capturas de pantalla y un repositorio público en GitHub.</p> <p>El sistema demostró eficiencia en la gestión de datos únicos, y permitió aplicar de forma práctica los conceptos teóricos de conjuntos y mapas en un contexto real.</p>
Conclusiones	<p>La implementación del sistema de registro de libros en una biblioteca permitió aplicar de forma práctica los conceptos de conjuntos y mapas, utilizando estructuras HashSet y Dictionary en lenguaje C#. Estas</p>

	<p>estructuras demostraron ser eficientes para garantizar la unicidad de los registros y facilitar la consulta rápida por código.</p> <p>Durante el desarrollo, se evidenció que:</p> <ul style="list-style-type: none"> • El uso de HashSet evitó la duplicación de códigos, asegurando integridad en el catálogo. • El Dictionary permitió acceder directamente a los títulos mediante claves únicas, optimizando el tiempo de búsqueda. • La interacción por consola, mediante un menú claro y funcional, facilitó la reportería básica solicitada en la guía. <p>Además, se contó con el acompañamiento de Microsoft Copilot como agente de apoyo técnico y ético, este agente contribuyó en la estructuración del código, la validación de la lógica condicional y la mejora de la claridad en los mensajes del sistema, sin reemplazar el criterio profesional de la estudiante.</p> <p>El sistema fue ejecutado correctamente en un entorno virtual, y se documentaron las evidencias mediante capturas de pantalla y un repositorio público en GitHub, se logró registrar más de 10 libros sin errores, demostrando la eficiencia y aplicabilidad de las estructuras utilizadas.</p> <p>Finalmente, esta práctica permitió fortalecer habilidades en programación estructurada, validación de datos, y documentación técnica, integrando teoría y práctica en un contexto funcional y ético.</p> <p>Para validar el funcionamiento del sistema, se realizó la ejecución con el registro de un libro, esta prueba fue suficiente para demostrar que la lógica de ingreso, consulta y visualización opera correctamente, y que el sistema está preparado para gestionar múltiples registros sin duplicación.</p>
Bibliografía	<p>Albahari, J., & Albahari, B. (2022). <i>C# 10 in a Nutshell: The Definitive Reference</i>. O'Reilly Media. (Referencia técnica sobre el lenguaje C# y sus estructuras de datos)</p> <p>Microsoft. (2023). <i>Queue<T> Clase</i> (System.Collections.Generic). Microsoft Learn. https://learn.microsoft.com/es-es/dotnet/api/system.collections.generic.queue-1 (Documentación)</p>

	<p><i>oficial sobre estructuras de cola en C#)</i></p> <p>Microsoft. (2023). <i>Dictionary<TKey,TValue> Clase</i> <i>(System.Collections.Generic). Microsoft Learn.</i> https://learn.microsoft.com/es-es/dotnet/api/system.collections.generic.dictionary-2 <i>(Referencia oficial sobre mapas y diccionarios en C#)</i></p> <p>Microsoft. (2023). <i>Copilot: Tu compañero de IA para programación y productividad.</i> https://www.microsoft.com/es-es/copilot <i>(Información sobre el agente de IA utilizado en la práctica)</i></p>
Anexos	<div data-bbox="611 904 1380 1503"></div> <p><i>Ilustración 1: Estudiante realizando Guía práctica</i></p>

```
main.cs
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
else
{
    foreach (var libro in catalogo)
    {
        Console.WriteLine($"Código: {libro.Key} | Título: {libro.Value}");
    }
}
break;

case 4:
    Console.WriteLine("Saliendo del sistema...");
    break;

default:
    Console.WriteLine("Opción inválida.");
    break;
}

} while (opcion != 4);
}
```

input

```
--- MENÚ BIBLIOTECA ---
1. Agregar libro
2. Consultar libro por código
3. Mostrar todos los libros
4. Salir
Seleccione una opción: 
```

Ilustración 2: Ingreso exitoso: ejecución de código

```
main.cs
1
2
3
4
5
6
7
8
9
10
using System;
using System.Collections.Generic;

class Biblioteca
{
    static void Main()
    {
        Dictionary<string, string> catalogo = new Dictionary<string, string>();
        HashSet<string> codigos = new HashSet<string>();
        int opcion;
    }
}
```

input

```
--- MENÚ BIBLIOTECA ---
1. Agregar libro
2. Consultar libro por código
3. Mostrar todos los libros
4. Salir
Seleccione una opción: 1
Ingrese código del libro: LIB002
Ingrese título del libro: EL PRINCIPITO
Libro agregado correctamente.

--- MENÚ BIBLIOTECA ---
1. Agregar libro
2. Consultar libro por código
3. Mostrar todos los libros
4. Salir
Seleccione una opción: 3

--- Catálogo de Libros ---
Código: LIB001 | Título:
Código: LIB002 | Título: EL PRINCIPITO
```

Ilustración 3: Probando la respuesta de la consola

Enlace de repositorio GitHub: <https://github.com/Sandruchi/Registro-de-Libros-CSharp.git>

Nombre estudiante: Sandra Maribel Rodríguez Rodríguez

Firma

Sandra R. R.