Universidad Mariano Gálvez de Guatemala

Ingeniería en Sistemas de Información y Ciencias de la Computación

Sede Virtual

ING. Walter Córdova

Programación lll

Sección 5

Examen Parcial 2

José Gabriel García Escobar

carne 7690-23-18731

10-05-2025

Serie I – Caminos en Árbol Binario con Suma Objetivo

Descripción del Problema

Se solicitó resolver un ejercicio en Java que consiste en encontrar todas las rutas de raíz a hoja en un árbol binario, donde la suma de los valores de los nodos sea igual a un número objetivo (targetSum). Una ruta válida es aquella que comienza en la raíz del árbol y termina en un nodo hoja (sin hijos).

Solución Implementada

Se implementó la clase Serie1 con el método pathSum, que resuelve el problema utilizando un enfoque recursivo con backtracking. Se recorre el árbol en profundidad (DFS) acumulando los valores de los nodos en una lista temporal. Si se llega a una hoja y la suma coincide con el targetSum, se guarda esa ruta en la lista final de resultados.

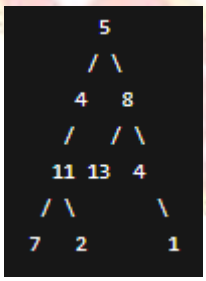
Principios Aplicados

Single Responsibility (SRP): Cada método tiene una responsabilidad clara. pathSum coordina la búsqueda y findPaths ejecuta la lógica recursiva.

Open/Closed Principle (OCP): La lógica es extensible para adaptarse a otras condiciones (como distintas reglas de suma), sin modificar la estructura básica del código.

Ejemplo Ejecutado

Con el siguiente árbol:



Y un targetSum = 22, el programa devuelve:

[[5, 4, 11, 2]]

Es decir, todas las rutas donde la suma de los nodos da como resultado 22.

Serie II

El proyecto tiene como propósito consumir datos públicos de una API de estadísticas COVID-19 (RapidAPI), procesarlos y guardarlos en una base de datos MySQL usando Java Spring Boot. Se aplican buenas prácticas de programación como principios SOLID, separación por capas y control de duplicados.

Componentes del Proyecto

1. Modelo (models)

RegionAPI.java: representa una región (ej. Europa, Asia).

ProvinciaAPI.java: representa una provincia (ej. Guatemala, Madrid).

ReporteAPI.java: contiene datos COVID como casos nuevos, activos, fallecidos, etc.

2. Repositorio (repositories)

- RepositorioRegion.java

Extiende JpaRepository. Permite guardar y consultar regiones.

- RepositorioProvincia.java

Permite persistir provincias asociadas a una región ISO.

- RepositorioReporte.java

Contiene métodos personalizados:

findByDateAndRegionIso(String date, String regionIso)

findByIsoAndDate(String iso, String date)

existsByDateAndIso(String date, String iso)

3. Servicio (services)

Clase central donde se implementa toda la lógica del proyecto:

- Descarga los datos desde la API (regiones, provincias, reportes).
- Transforma la respuesta JSON usando ObjectMapper.
- Guarda los datos en la base con validaciones.

- Implementa control para evitar duplicados.

Usa TreeMap para agrupar los reportes por provincia (clave: nombre de la provincia) y conservar solo el último registro en caso de duplicados.

CONTROLADOR (controllers)

MainController.java

Define todos los endpoints disponibles, entre ellos:

/api/ejecutar: Ejecuta un hilo que descarga y guarda datos solo si aún no existen.

/api/regiones, /api/provinciaapi, /api/reportes: Descarga manual de cada recurso.

/api/regiones/ver, /api/provinciaapi/ver, /api/reportes/ver: Visualización de los datos guardados.

/api/reportes/{iso}/{date}: Devuelve reportes por país y fecha.

/api/reportes/agrupados?iso=GT&date=2022-05-01: (NUEVO – SERIE II) Agrupa los reportes por provincia y elimina duplicados.□ Cómo Probar en Postman

LO NUEVO IMPLEMENTADO EN LA SERIE II

Control de ejecuciones:

Se verifica en base de datos si ya se descargaron reportes para cierto país (iso) y fecha (date) antes de ejecutar el hilo.

Agrupación y eliminación de duplicados:

Se implementó un endpoint nuevo (/api/reportes/agrupados) que usa TreeMap para mostrar solo un reporte por provincia, evitando datos repetidos.

Nuevos métodos en ServicioPeticion y RepositorioReporte:

obtenerReportesUnicosPorProvincia(String iso, String date) fue agregado.

Se reusaron métodos como findByIsoAndDate() para filtrar la base.

Iniciar el hilo automático

Descripción: Lanza un hilo que descarga regiones, provincias y reportes si no existen.

Descargar datos manualmente

Regiones:

GET http://localhost:8080/api/regiones

Provincias:

GET http://localhost:8080/api/provinciaapi

Reportes:

GET http://localhost:8080/api/reportes

Ver regiones:

GET http://localhost:8080/api/regiones/ver

Ver provincias:

GET http://localhost:8080/api/provinciaapi/ver

Ver todos los reportes:

GET http://localhost:8080/api/reportes/ver

🔍 Ver reportes por país (ISO) y fecha

GET http://localhost:8080/api/reportes/{iso}/{date}

Ejemplo:

GET http://localhost:8080/api/reportes/GT/2022-05-01

Ver reportes sin duplicados por provincia (agrupados)

GET http://localhost:8080/api/reportes/agrupados?iso=GT&date=2022-05-01

Requisitos para correr

Tener MySQL corriendo

Crear base de datos y configurar application.properties

Tener Java 17 y Maven instalados

Nota agregue solo “/api” para mejor practica que en el video no sale.