UNIVERSIDAD DE SAN BUENAVENTURA CALI

Facultad de Ingeniería

Programa de Ingeniería de Sistemas

Asignatura: Técnicas Avanzadas de Programación

Reporte de seguimiento a metodología ADIM - Figuras 2D y 3D

Integrantes:

Carrera Medina Sebastian - Código: 75298

Drada Alarcon Valeria - Código: 132001

Torres Patino Jose Manuel - Código: 130053

Docente:

Cabezas Troyano Ivan Mauricio

Fecha:

30 Julio de 2025

Introducción

Este informe presenta el desarrollo de un sistema para el cálculo de propiedades geométricas de figuras en dos y tres dimensiones, aplicado bajo la metodología ADIM (Análisis, Diseño, Implementación y Mantenimiento).

El objetivo principal del proyecto es aplicar los conceptos de la Programación Orientada a Objetos (POO) utilizando diferentes lenguajes de programación: Go, Python y Java, implementados de forma individual por los integrantes del equipo.

Cada versión del sistema permite al usuario interactuar mediante un menú en la línea de comandos (CLI), donde puede seleccionar figuras geométricas (como Cuadrado, Círculo, Cubo y Esfera) y obtener cálculos como área, perímetro o volumen, según corresponda.

A través de este trabajo se buscó reforzar la modularidad del código, el diseño con clases, el uso de interfaces, y el enfoque estructurado de desarrollo de software, asegurando además la escalabilidad del sistema para incluir futuras figuras y funcionalidades.

Introducción **2**

1. Analisis **4**

2. Diseño **4**

2. Diagramas de clases UML: **7**

3. Implementación **8**

4. Mantenimiento **9**

5. Conclusion **9**

Reporte de seguimiento a Metodología ADIM

Proyecto: Sistema para el cálculo de propiedades de figuras geométricas 2D y 3D.

Lenguaje utilizados: Go, Python y Java.

1. Analisis

a. Problema identificado:

Existe la necesidad de calcular de forma dinámica el área, el perímetro o el volumen de diferentes figuras geométricas, clasificadas en 2D y 3D, de manera sencilla y directa desde un sistema de consola.

b. Requisitos del sistema:

- **i.** El sistema debe ejecutarse desde la línea de comandos(CLI)
- **ii.** El sistema debe presentar un menú de navegación con submenús para figuras 2D y 3D
- **iii.** El sistema debe calcular el área y perimétrico para las figuras 2D
- **iv.** El sistema debe calcular el volumen para las figuras 3D

c. Figuras implementadas:

- **i.** 2D: Cuadrado, Círculo
- **ii.** 3D: Cubo, Esfera.

d. Restricciones

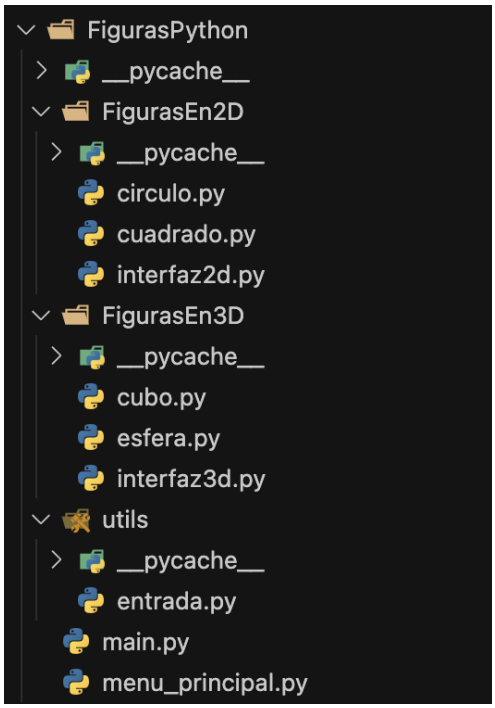
- **i.** El sistema debe desarrollarse aplicando **Programación Orientada a Objetos (POO)** y siguiendo principios **SOLID**
- **ii.** El desarrollo debe implementarse en **tres lenguajes**, siendo **Python** y **Java** obligatorios.
- **iii.** El proyecto debe seguir la metodología **ADIM**.

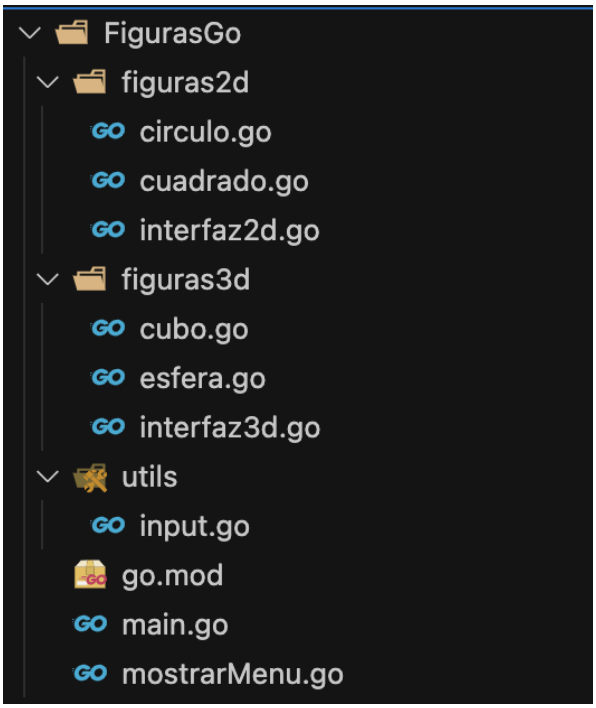
2. Diseño

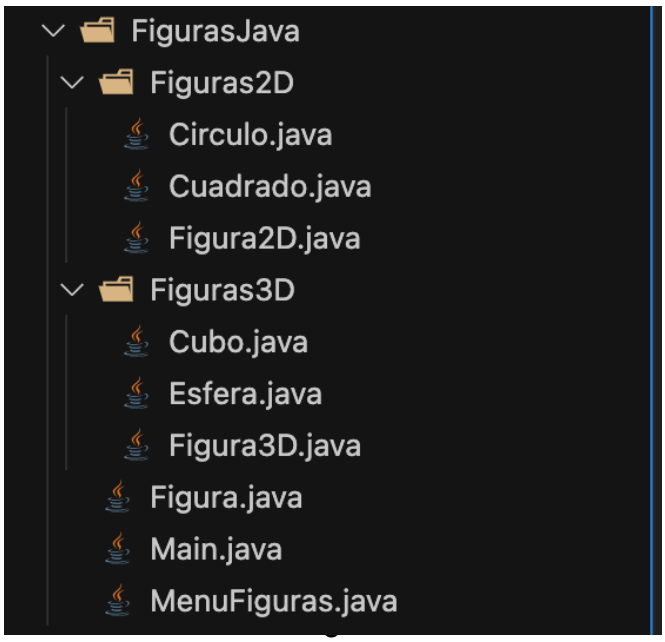
a. Paradigma: Programación Orientada a objetos(POO) en Go, Python y Java.

b. Organización modular del código:

i. Go y Python



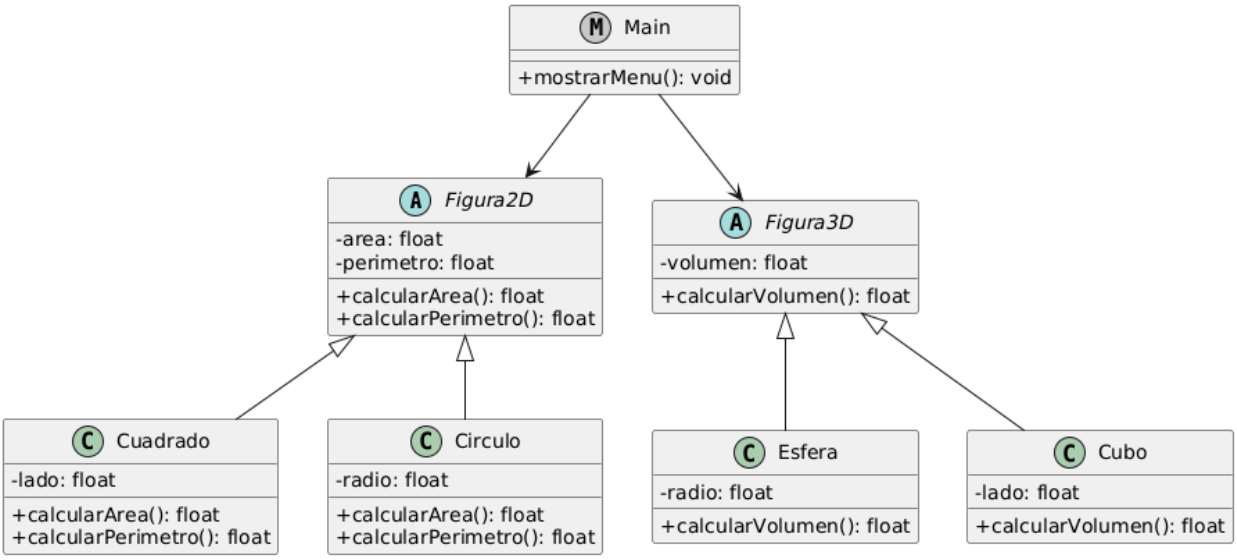




Archivos/carpetas	Descripción
main.go main.py main.java	Archivo principal del proyecto. Solo contiene el punto de entrada del programa y llama a la función `MostrarMenu()`para iniciar la interacción.
figuras2d/cuadrado.go figurasEn2d/cuadrado.py figurasEn2d/cuadrado.java	Implementa el cálculo de área y perímetro para un cuadrado.
figuras2d/circulo.go figurasEn2D/circulo.py figurasEn2D/circulo.java	Implementa el cálculo de área y perímetro para un círculo.
figuras3d/cubo.go figurasEn3D/cubo.py figurasEn3D/cubo.java	Implementa el cálculo de volumen de un cubo.
figuras3d/esfera.go figurasEn3D/esfera.py figurasEn3D/esfera.java	Implementa el cálculo de volumen de una esfera.
utils/input.go utils/input.py utils/input.java	Contiene funciones para leer valores numéricos del usuario.
figuras2d/interfaz2d.go figuras3d/interfaz3d.go figuras3d/interfaz3d.java	Define la interfaz `Figura2D`, que establece los métodos comunes para figuras bidimensionales (área y perímetro) y la interfaz `Figura3D`, que establece el método común para figuras tridimensionales (volumen).
mostrarMenu.go Mostrar_menu.py mostrar_menu.java	Contiene la lógica del menú principal y submenús (2D y 3D), permitiendo al usuario seleccionar figuras y realizar cálculos de área, perímetro o volumen.

Archivos/carpetas	Descripción
main.go main.py main.java	Archivo principal del proyecto. Solo contiene el punto de entrada del programa y llama a la función `MostrarMenu()` (Go) o `mostrar_menu()` (Python) para iniciar la interacción.
go.mod	Archivo del módulo de Go para manejo de dependencias.

2. Diagramas de clases UML:



3. Implementación

a. El proyecto fue desarrollado de forma colaborativa, donde cada integrante implementó la misma lógica de figuras geométricas (2D y 3D) en un lenguaje diferente, respetando el diseño basado en Programación Orientada a Objetos (POO) y el diagrama de clases UML compartido.

b. Aunque los lenguajes son diferentes, se implementaron los mismos métodos clave:

i. **CalcularArea(), CalcularPerimetro()** para figuras 2D.

ii. **CalcularVolumen()** para figuras 3D

iii. En Java, además se usaron **getNombre(), setNombre(), setLado()**, **getLado()**, **setRadio()**, **getRadio()** según corresponda a la figura.

c. Cada versión permite la ejecución desde línea de comandos mediante un menú interactivo que permite al usuario seleccionar entre figuras 2D o 3D y luego realizar los cálculos respectivos.

d. La estructura modular en los tres lenguajes permite mantener el código organizado, legible y fácilmente extensible.

4. Mantenimiento

a. La estructura modular permite agregar nuevas figuras fácilmente (por ejemplo. Triangulo, Cilindro)

b. La separación en archivos por figura mejora la legibilidad y escalabilidad del código.

c. El uso de interfaces (`Figura2D`, `Figura3D`) permite extender funcionalidades sin modificar el código existente, facilitando el mantenimiento y cumpliendo con el principio Open/Closed de SOLID.

d. Las funciones centralizadas para entrada de datos (`LeerFloat`, `LeerEntero`) permiten validar todos los valores desde un único lugar, facilitando su modificación o mejora futura.

5. Conclusion

El desarrollo del sistema permitió aplicar los conceptos fundamentales de la Programación Orientada a Objetos (POO), incluyendo herencia, abstracción e interfaces, en tres lenguajes distintos: Go, Python y Java. Se logró cumplir con los lineamientos del curso, incluyendo la metodología ADIM y la ejecución desde CLI mediante un menú interactivo.

Además, la organización modular del código y la implementación de principios SOLID favorecieron la extensibilidad del sistema, facilitando la incorporación de nuevas figuras en el futuro.

El trabajo colaborativo por lenguajes también reforzó la comprensión de cómo aplicar los mismos principios en diferentes entornos de programación.