PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

PROGRAMACIÓN 3

4ta. práctica (tipo b)
(Segundo Semestre 2025)

Indicaciones Generales:

- Tiempo estimado: 1h 50 minutos
- Se les recuerda que, de acuerdo al reglamento disciplinario de nuestra institución, constituye una falta grave copiar del trabajo realizado por otro estudiante o cometer plagio para el desarrollo de esta práctica.
- Para el desarrollo de toda la práctica debe utilizar el sistema operativo **Windows**, así como el entorno de desarrollo integrado **APACHE NETBEANS 27** y el **JDK 24**.
- Está permitido el uso de apuntes de clase, diapositivas, ejercicios de clase y código fuente.
  (Debe descargarlos antes de iniciar con la solución del enunciado)
- Está permitido el uso de Internet (únicamente para consultar páginas oficiales de Microsoft y Oracle). No obstante, está prohibida toda forma de comunicación con otros estudiantes o terceros.

PARTE PRÁCTICA (20 puntos)

PUEDE UTILIZAR MATERIAL DE CONSULTA.
Antes de comenzar el laboratorio, descargue todos los proyectos, apuntes, diapositivas que utilizará.
Se considerará en la calificación el uso de buenas prácticas de programación (aquellas vistas en clase).

PREGUNTA 1: Creación de un motor de base de datos MySQL en AWS Academy (3.5 puntos)

Se le solicita ingresar al servicio de AWS Academy (https://www.awsacademy.com/vforcesite/LMS_Login) y crear una instancia o motor de base de datos MySQL con los siguientes parámetros:

- Método de creación de base de datos: **Creación estándar**
- Tipo de motor: **MySQL**
- Edición: **Comunidad de MySQL**
- Versión del motor: **MySQL 8.0.43**
- Plantilla: **Entorno de pruebas**
- Opciones de implementación: **Implementación de una instancia de base de datos de zona de disponibilidad única (1 instancia)**
- Identificador de instancias de base de datos: **labs-2025-2-prog3**
- Nombre de usuario maestro: **root**
- Contraseña maestra: **prog31inf30**
- Clase de instancia de base de datos: **db.tg4.micro**
- Capacidad de disco duro para la BD: **20 GiB SSD (disco sólido) sin posibilidad de escalado automático**
- **Acceso público: SI**
- **Puerto de la base de datos: 3306**
- Nombre de base de datos inicial: **laboratorio4**
- Habilitar las copias de seguridad automatizadas: **NO**
- Habilitar el cifrado: **NO**

- **Configure las reglas de entrada para que cualquier IP o computador tenga acceso a su base de datos.**
- **IMPORTANTE: El motor de base de datos debe ser configurado para que cualquier computador o IP pueda acceder al mismo. Si esta configuración no es realizada y no es posible conectarse a su motor de base de datos desde la computadora de los jefes de práctica o docente, entonces NO se considerará puntaje alguno.**

Una vez creada su base de datos, se le solicita comprobar que funciona correctamente con el **MySQL Workbench**. Se le solicita dejar la base datos **PRENDIDA** o **ACTIVA** en AWS y colocar los datos de conexión en un archivo llamado **Pregunta01_códigoAlumno.txt** (en el archivo deberá colocar **el *hostname***, **el *username***, **el *password*** y **el nombre de la BD** o **esquema**). Compruebe que es posible acceder a su base de datos creada y **suba el archivo .txt a PAIDEIA.**

PREGUNTA 2: Ejecución de un Script SQL en la base de datos creada (1.5 punto)

Descargue el script SQL que se encuentra en PAIDEIA y ejecute el mismo en la base de datos creada.
Ejecute la instrucción **SELECT * FROM videojuego;** para verificar que se ha creado correctamente la tabla y los registros.

PREGUNTA 3: Ejercicio de generación de JAR (5.0 puntos)

El grupo de investigación **AVATAR** de nuestra casa de estudios, enfocado en potenciar procesos educativos, sociales y culturales a través del desarrollo de videojuegos ha implementado en JAVA las clases que se muestran en la Figura 01, las cuales se encuentran organizadas en los paquetes que se muestran.

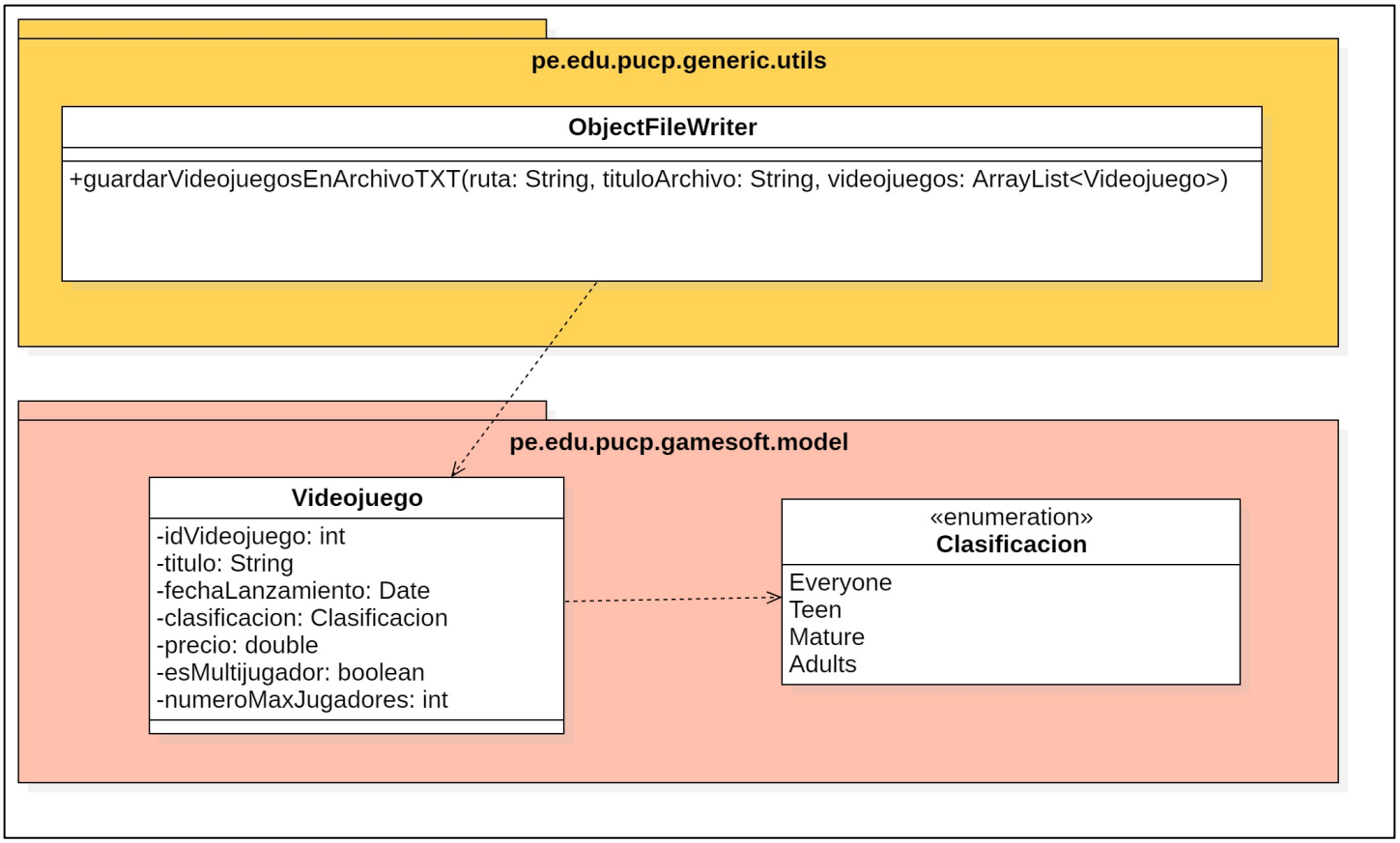

Fig. 01. Diagrama de clases

La implementación las clases permiten almacenar en un archivo de texto plano, la información de una lista de objetos de tipo "Videojuego". Se le solicita descargar el código fuente de esta implementación desde PAIDEIA y crear un ensamblado JAR que contenga toda esta estructura con las tres clases respetando los paquetes definidos para cada una de las mismas. La creación de la librería debe realizarse mediante líneas de comandos y debe guardar en un archivo de texto (.txt) **TODAS** las instrucciones que ha ejecutado por consola para la creación de la librería. El ensamblado debe llamarse "**utilitarios.jar**".

Recuerde que un archivo .JAR contiene archivos compilados de tipo .class por lo que la primera instrucción será compilar primero todos los archivos fuente. Esta instrucción también debe adjuntarse en el archivo .txt

Antes de ejecutar cualquier instrucción, ejecute lo siguiente, que permitirá configurar la misma versión de JAVA que está utilizando el NETBEANS:

```
SET PATH=C:\Program Files\Apache NetBeans\jdk\bin;%PATH%
```

Para generar un ensamblado JAR, la instrucción es la siguiente:

```
jar cvf <nombre del archivo JAR a generar> <ruta de los archivos .class>
```

Por ejemplo, si el archivo JAR a generar fuese "**ejemplo.jar**" y los archivos .class se encuentran en la ruta: "**com/company/domain**", la instrucción sería la siguiente:

```
jar cvf ejemplo.jar com/company/domain/*.class
```

Suba el archivo .txt que contiene todas las instrucciones utilizadas por consola, así como el archivo creado "**utilitarios.jar**". Se verificará que este JAR haya sido creado por líneas de comandos y no mediante el IDE.

PREGUNTA 4: Uso del JAR en un proyecto NETBEANS (5.0 puntos)
Se le solicita descargar el proyecto JAVA de NETBEANS desde PAIDEIA y copiar el archivo JAR generado en la pregunta anterior en el paquete "**pe.edu.pucp.gamesoft.lib**".
Debe quedar como se muestra en la Figura 02.

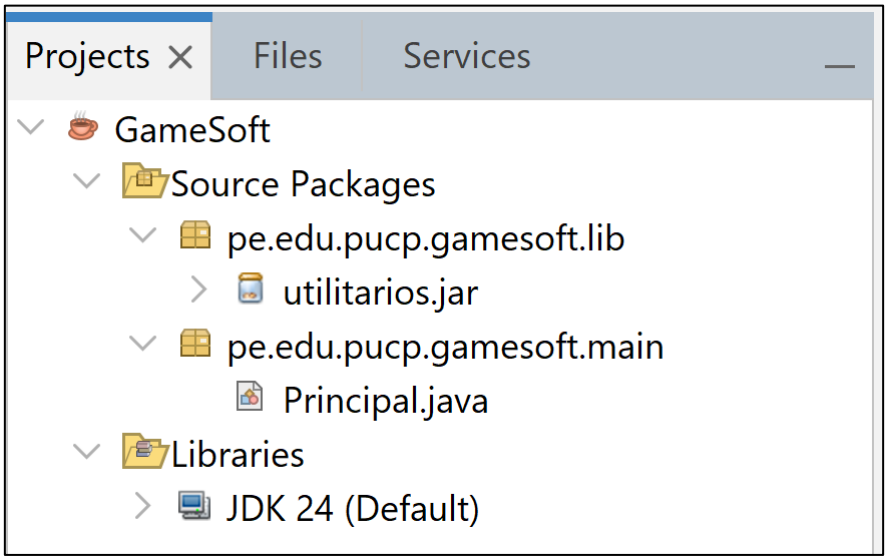

Fig. 02. Estructura del proyecto

Ahora se le solicita dar clic derecho a "**Libraries**" y dar clic a "**Add JAR/Folder**" como en la **Figura 03**.

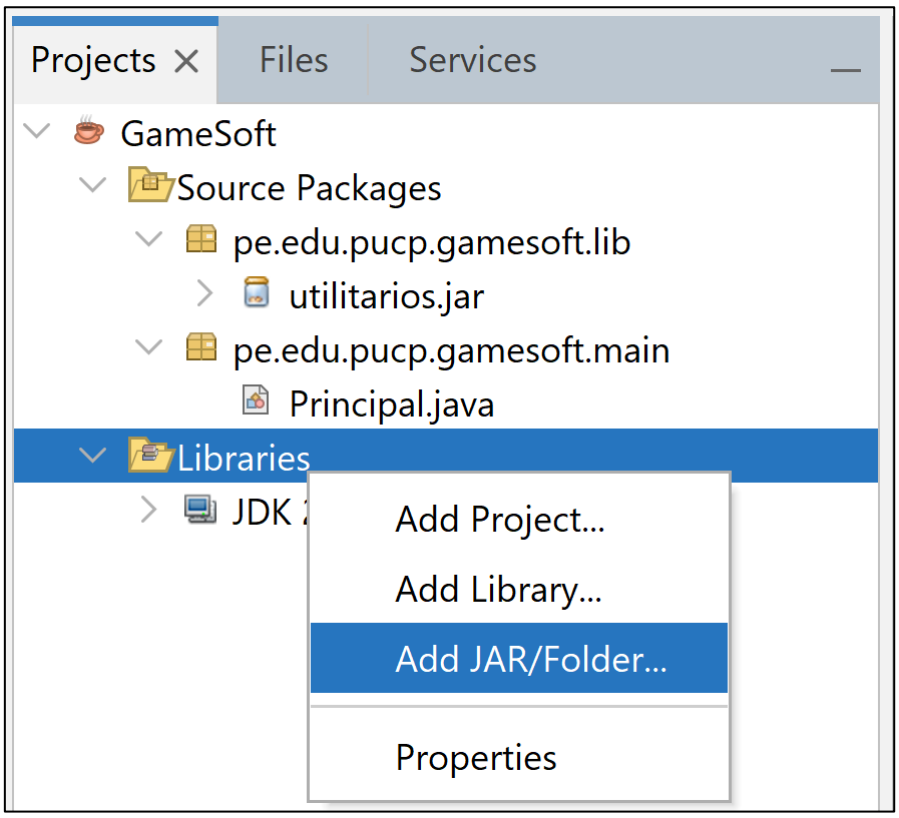

Fig. 03. Agregando referencia JAR al proyecto

Ubique el archivo "utilitarios.jar" que se encuentra DENTRO del proyecto en la carpeta:
"**GameSoft\src\pe\edu\pucp\gamesoft\lib"**
Agregue la referencia y asegúrese de que esté seleccionada "**ruta relativa**".

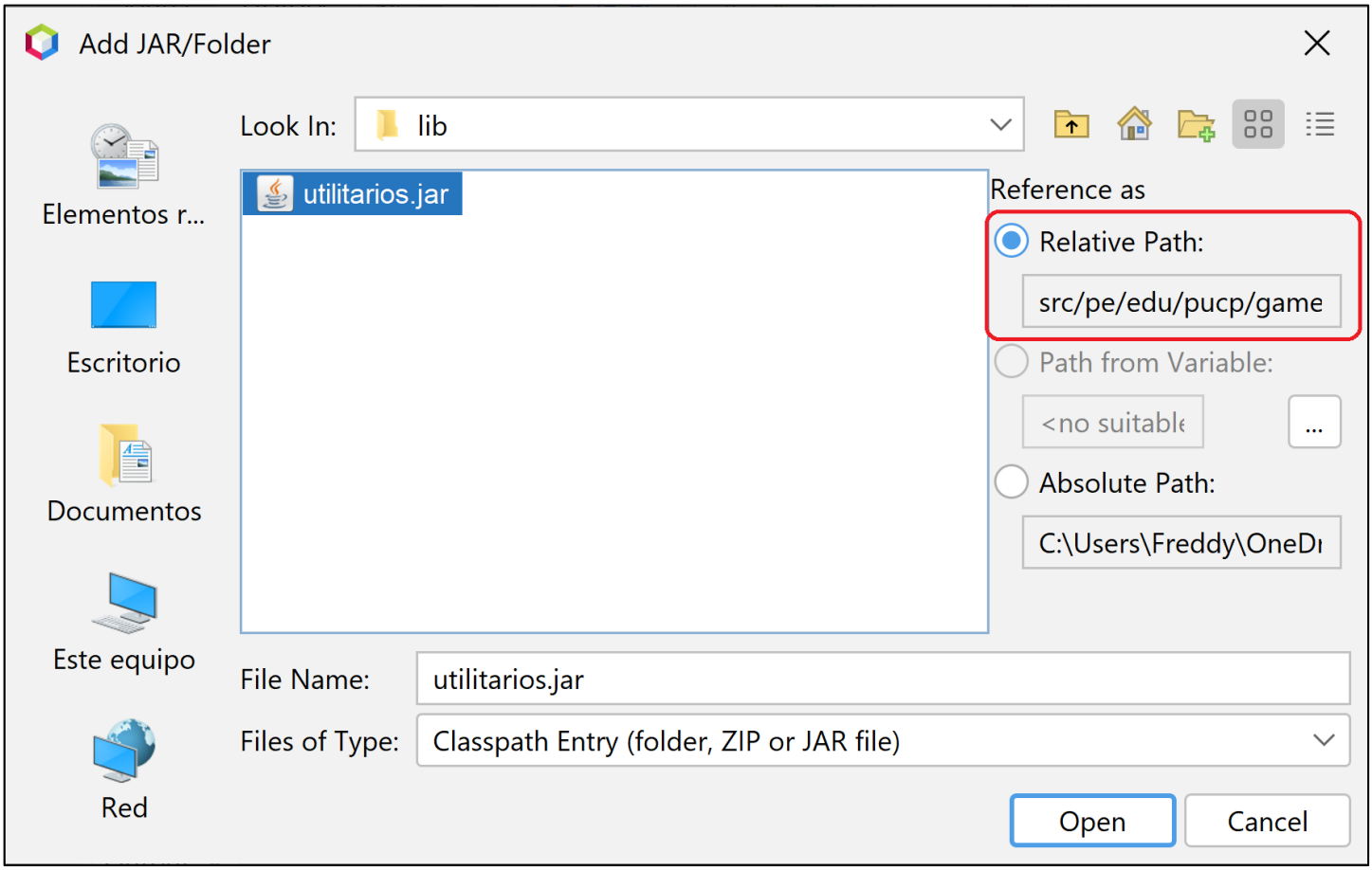

Fig. 04. Selección de archivo JAR con ruta relativa

La estructura del proyecto debe quedar como en la Figura 05.
Ahora agregamos las líneas de código en JAVA necesarias en el método main() de la clase Principal para probar el JAR en nuestro programa. Agregue las líneas de código que se muestran en la Figura 06 y verifique que se guardan los objetos en el archivo .txt.

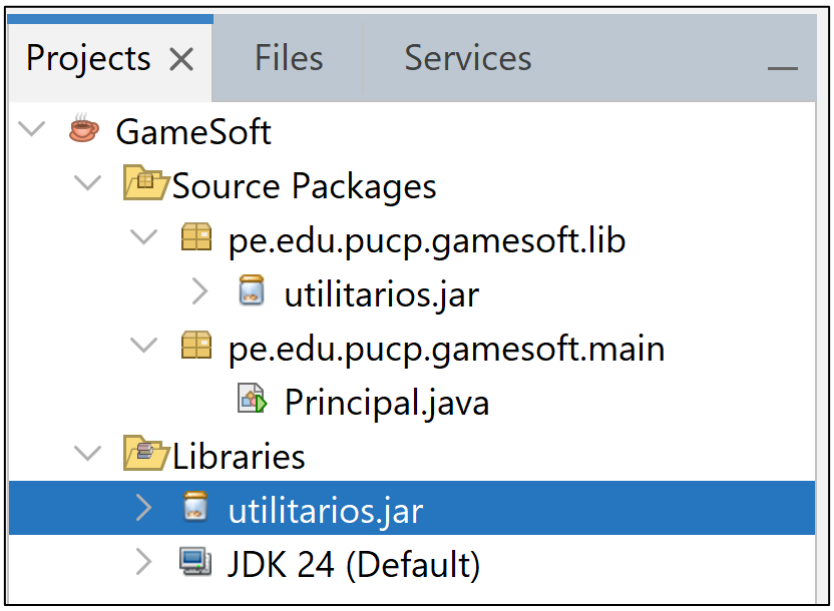

Fig. 05. Estructura final del proyecto solicitado

```
package pe.edu.pucp.gamesoft.main;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import pe.edu.pucp.gamesoft.model.Clasificacion;
import pe.edu.pucp.gamesoft.model.Videojuego;
```

```java
import pe.edu.pucp.generic.utils.ObjectFileWriter;
import java.util.Date;
public class Principal {
    public static void main(String[] args){
        // Creamos una lista de videojuegos
        ArrayList<Videojuego> lista = new ArrayList<>();

        //Creamos un SimpleDateForma para manejo de fechas
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

        //Creamos las fechas
        Date fechaLazv1 = null;
        Date fechaLazv2 = null;
        Date fechaLazv3 = null;
        try{
            fechaLazv1 = sdf.parse("2017-02-03");
            fechaLazv2 = sdf.parse("2017-03-28");
            fechaLazv3 = sdf.parse("2020-05-19");
        }catch(ParseException ex){
            System.out.println("ERROR: " + ex.getMessage());
        }

        // Creamos algunos objetos videojuego
        Videojuego v1 = new Videojuego("Zelda: Breath of the Wild", fechaLazv1,
Clasificacion.Teen, 59.99, false, 1);
        Videojuego v2 = new Videojuego("Mario Kart 8 Deluxe", fechaLazv2, Clasificacion.Everyone,
49.99, true, 4);
        Videojuego v3 = new Videojuego("The Last of Us Part II", fechaLazv3, Clasificacion.Mature,
69.99, false, 1);

        // Agregamos a la lista
        lista.add(v1);
        lista.add(v2);
        lista.add(v3);

        // Creamos escritor y guardamos en archivo
        ObjectFileWriter writer = new ObjectFileWriter();
        writer.guardarVideojuegosEnArchivoTXT("T:\\", "videojuegos.txt", lista);
    }
}
```

Fig. 06. Contenido de la clase Principal

<u>PREGUNTA 05</u>: Conexión a Base de Datos y Registro (5.0 puntos)

Ahora agregue a la clase Principal el método definido en la Figura 07 y agregue las líneas necesarias a lo programado en la pregunta anterior para que sea posible, además de la generación del archivo de texto, el registro en la base de datos.

```java
public static void guardarVideojuegosEnBD(ArrayList<Videojuego> listaVideojuegos) {
        String sql = "INSERT INTO videojuego (titulo, fecha_lanzamiento, clasificacion, precio,
es_multijugador, numero_max_jugadores) VALUES (?, ?, ?, ?, ?, ?)";

    try (Connection conn = DriverManager.getConnection(URL, USER, PASSWORD)) {
        PreparedStatement ps = conn.prepareStatement(sql);

        for (Videojuego vj : listaVideojuegos) {
            ps.setString(1, vj.getTitulo());
            // Convertimos de java.util.Date a java.sql.Date
            if (vj.getFechaLanzamiento() != null) {
                ps.setDate(2, new java.sql.Date(vj.getFechaLanzamiento().getTime()));
            } else {
                ps.setDate(2, null);
            }
            ps.setString(3, vj.getClasificacion().toString());
            ps.setDouble(4, vj.getPrecio());
            ps.setBoolean(5, vj.isEsMultijugador());
```

```
            ps.setInt(6, vj.getNumeroMaxJugadores());

            ps.executeUpdate();
        }

        System.out.println("Videojuegos guardados en la BD correctamente.");
    } catch (SQLException e) {
        System.err.println("Error al guardar en BD: " + e.getMessage());
    }
}
```

Fig. 07. Método que permite la conexión y el registro en la base de datos.

Recuerde que debe definir las variables URL, USER y PASSWORD de acceso al motor de base de datos que ha creado en la Pregunta 01. Asimismo, debe realizar las importaciones que son requeridas y agregar el “mysql-connector-j-9.4.0.jar” siguiendo exactamente el mismo procedimiento realizado para “utilitarios.jar”.
Suba el proyecto final a PAIDEIA.

Profesores:

- **Dr. Freddy Paz**
- **Dr. Andrés Melgar**
- **Mag. Eric Huiza**

10 de setiembre del 2025