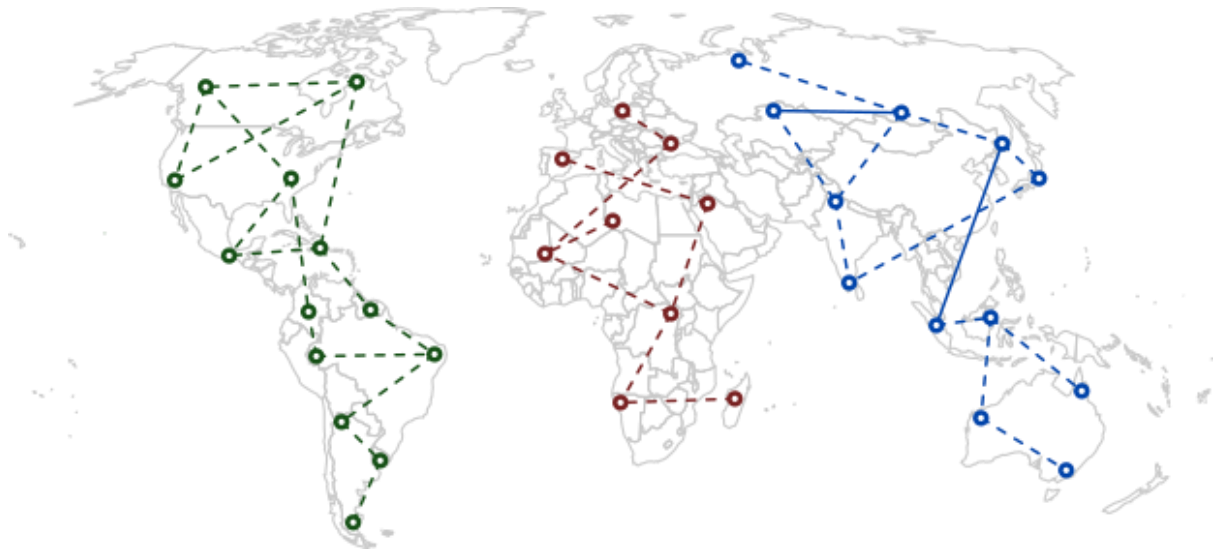


<b>EVALUACION</b>	Obligatorio	<b>GRUPO</b>	Todos	<b>FECHA</b>	Agosto 2024 – V1
<b>MATERIA</b>	Algoritmos y Estructuras de Datos 2				
<b>CARRERA</b>	Analista Programador – Analista en TI				
<b>CONDICIONES</b>	<p>- <b>Puntaje máximo:</b> 35 puntos</p> <p>- <b>Puntaje mínimo:</b> 0 puntos</p> <p>- <b>Fecha de entrega:</b> 07/11/2024 hasta las 21:00 horas en <a href="https://gestion.ort.edu.uy">gestion.ort.edu.uy</a> (max. 40Mb en formato zip, rar o pdf)</p> <p><b>Uso de material de apoyo y/o consulta</b></p> <p><u>Inteligencia Artificial Generativa</u></p> <ul style="list-style-type: none"> <li>- Seguir las pautas de los docentes: Se deben seguir las instrucciones específicas de los docentes sobre cómo utilizar la IA en cada curso.</li> <li>- Citar correctamente las fuentes y usos de IA: Siempre que se utilice una herramienta de IA para generar contenido, se debe citar adecuadamente la fuente y la forma en que se utilizó.</li> <li>- Verificar el contenido generado por la IA: No todo el contenido generado por la IA es correcto o preciso. Es esencial que los estudiantes verifiquen la información antes de usarla.</li> <li>- Ser responsables con el uso de la IA: Conocer los riesgos y desafíos, como la creación de “alucinaciones”, los peligros para la privacidad, las cuestiones de propiedad intelectual, los sesgos inherentes y la producción de contenido falso</li> <li>- En caso de existir dudas sobre la autoría, plagio o uso no atribuido de IAG, el docente tendrá la opción de convocar al equipo de obligatorio a una defensa específica e individual sobre el tema</li> </ul> <p><b>IMPORTANTE:</b></p> <p>1) Inscribirse.</p> <p>2) Formar grupos de hasta 2 personas del mismo dictado.</p> <p>3) Subir el trabajo a Gestión antes de la hora indicada (ver hoja al final del documento: “RECORDATORIO”)</p> <p>Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con el Coordinador o Coordinación adjunta <b>antes de las 20:00hs.</b> del día de la entrega, a través de los mails <a href="mailto:alamon@ort.edu.uy">alamon@ort.edu.uy</a> y <a href="mailto:fernandez_ma@ort.edu.uy">fernandez_ma@ort.edu.uy</a>, o telefónicamente al 29021505 - int 1156 u 1138</p>				

## Introducción

Se desea implementar un programa para ingresar y consultar información sobre los jugadores, equipos y sucursales de una empresa de videojuegos que organiza torneos. Cada operación tiene restricciones de tiempo específicas de acuerdo con las necesidades de la empresa.



Todas las operaciones deberán devolver una instancia de la clase Retorno. Dicha clase contiene:

- Un **resultado**, que especifica si la operación se pudo realizar correctamente (OK), o si ocurrió algún error (según el número de error).
- Un **valorEntero**, para las operaciones que retornen un número entero.
- Un **valorString**, para las operaciones que retornen un String, o un valor más complejo, por ejemplo, un listado, el cual será formateado según lo indicado en los ejemplos.

```
public class Retorno {
    public enum Resultado {
        OK,
        ERROR_1, ERROR_2, ERROR_3, ERROR_4, ERROR_5, ERROR_6, ERROR_7,
        NO_IMPLEMENTADA
    }

    private Resultado resultado;
    private final int valorEntero;
    private String valorString;

    . . .
}
```

Se provee: una interfaz llamada **Sistema**, la cual no podrá ser modificada en ningún sentido, y una clase **ImplementacionSistema** que la implementa, donde su equipo deberá completar la implementación de las operaciones solicitadas.

Consideraciones:

- La clase `ImplementacionSistema` NO PODRÁ SER UN SINGLETON. Debe ser una clase instanciable.
- Pueden definirse tipos de datos (clases) auxiliares.
- Se provee un proyecto base con la estructura de las clases.

## Funcionalidades

### 01. Inicializar Sistema

Retorno `inicializarSistema(int maxSucursales);`

Descripción: Inicializa las estructuras necesarias para representar el sistema especificado, capaz de registrar como máximo la cantidad `maxSucursales` de sucursales diferentes en el sistema.

Restricción de eficiencia: no tiene.

Retornos posibles	
OK	Si el sistema fue inicializado exitosamente.
ERROR	1. Si <code>maxSucursales</code> es menor o igual a 3.
NO_IMPLEMENTADA	Cuando aún no se implementó. Es el tipo de retorno por defecto.

## 02. Registrar Jugador

Retorno **registrarJugador**(String alias, String nombre, String apellido, Categoria categoria);

Descripción: Registra el jugador con sus datos. El alias es su identificador único.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio siendo  $n$  la cantidad total de jugadores.

Retornos posibles	
OK	Si el jugador fue registrado exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros es vacío o <i>null</i>.</li> <li>2. Si ya existe un jugador registrado con ese alias.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Las categorías posibles son “Principiante”, “Estándar” y “Profesional”.

## 03. Buscar Jugador

Retorno **buscarJugador**(String alias);

Descripción: Retorna en el valorString del retorno los datos del jugador con el siguiente formato “alias;nombre;apellido;categoria”. Además, en el campo valorEntero de la clase Retorno, deberá devolver la cantidad de elementos que recorrió durante la búsqueda en la estructura utilizada.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio siendo  $n$  la cantidad total de jugadores.

Retornos posibles	
OK	<p>Si el jugador se encontró.</p> <p>Retorna en <i>valorString</i> los datos del jugador.</p> <p>Retorna en <i>valorEntero</i> la cantidad de elementos recorridos durante la búsqueda.</p>
ERROR	<ol style="list-style-type: none"> <li>1. Si el alias es vacío o <i>null</i>.</li> <li>2. Si no existe un jugador registrado con ese alias.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 04. Listar jugadores por alias ascendente

Retorno **listarJugadoresAscendente()**;

Descripción: Retorna en *valorString* los datos de todos los jugadores registrados, ordenados por alias en forma creciente.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(n)** siendo n la cantidad total de jugadores.

Retornos posibles	
OK	Retornando el listado de jugadores en <i>valorString</i> .
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:

alias1;nombre1;apellido1;categoria1|alias2;nombre2;apellido2;categoria2

## 05. Listar jugadores por categoría

Retorno **listarJugadoresPorCategoria(Categoria unaCategoria)**;

Descripción: Retorna en *valorString* los datos de todos los jugadores registrados con esa categoría ordenados de forma creciente por alias.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(k)**, siendo k la cantidad de jugadores con dicha categoría.

Retornos posibles	
OK	Si se pudo listar los jugadores que pertenecen a esa categoría correctamente.
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno de *valorString*:

alias1;nombre1;apellido1;categoria|alias2;nombre2;apellido2;categoria

## 06. Registrar equipo

Retorno **registrarEquipo**(String nombre, String manager);

Descripción: Registra el equipo en el sistema indicando su nombre (identificador único) y el nombre del mánager.

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n)$**  promedio siendo n la cantidad total de equipos.

Retornos posibles	
OK	Si el equipo fue registrado exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros es vacío o null.</li> <li>2. Si ya existe un equipo con ese nombre.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 07. Agregar jugador a equipo

Retorno **agregarJugadorAEquipo**(String nombreEquipo, String aliasJugador);

Descripción: Agrega el jugador al equipo indicado, un equipo puede tener un máximo de cinco integrantes. Solo se puede agregar jugadores que tengan la categoría "Profesional".

Restricción de eficiencia: Esta operación deberá realizarse en orden  **$O(\log n) + O(\log m)$**  promedio siendo n la cantidad total de equipos y m la cantidad total de jugadores.

Retornos posibles	
OK	Si el jugador fue agregado correctamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si alguno de los parámetros es vacío o null.</li> <li>2. Si no existe un equipo con ese nombre.</li> <li>3. Si no existe un jugador con ese alias.</li> <li>4. Si el equipo ya tiene 5 integrantes.</li> <li>5. Si el jugador no tiene la categoría profesional.</li> <li>6. Si el jugador ya pertenece a otro equipo.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 08. Listar los jugadores de un equipo.

Retorno **listarJugadoresDeEquipo**(String nombreEquipo);

Descripción: Retorna en valorString los datos de todos los jugadores pertenecientes al equipo ordenados de forma creciente por alias.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(log n)** promedio, siendo n la cantidad total de equipos.

Retornos posibles	
OK	Si se pudo listar los jugadores que pertenecen a dicho equipo.
ERROR	<ol style="list-style-type: none"> <li>1. Si el nombre es vacío o null.</li> <li>2. Si no existe un equipo con ese nombre.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno de *valorString*:

alias1;nombre1;apellido1;Profesional|alias2;nombre2;apellido2;Profesional

## 09. Listar equipos por nombre descendente

Retorno **listarEquiposDescendente**();

Descripción: Retorna en valorString los datos de todos los equipos ordenados por nombre en forma decreciente.

Restricción de eficiencia: Esta operación deberá realizarse en orden **O(n)** siendo n la cantidad total de equipos.

Retornos posibles	
OK	Retornando el listado de equipos en <i>valorString</i> .
ERROR	No hay errores posibles.
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:

nombreEquipo2;nombreManager2;cantidadJugadores2|  
nombreEquipo1;nombreManager1;cantidadJugadores1;

Por ejemplo:

FuSion;Fabián;3|Aeon;Ana;4

## 10. Registrar sucursal

Retorno **registrarSucursal**(String codigo, String nombre);

Descripción: Registra la sucursal en el sistema con el código y nombre indicado. El código es el identificador único, el código y nombre no pueden ser vacíos.

Retornos posibles	
OK	Si la sucursal fue registrada exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si en el sistema ya hay registrados <i>maxSucursales</i>.</li> <li>2. Si código o nombre son vacíos o null.</li> <li>3. Si ya existe una sucursal con ese código.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Esta operación no tiene restricciones de eficiencia.

## 11. Registrar Conexión

Retorno **registrarConexion**(String codigoSucursal1, String codigoSucursal2, int latencia);

Descripción: Registra una conexión entre dos sucursales y la latencia de dicha conexión, la conexión es bidireccional.

Retornos posibles	
OK	Si la conexión fue registrada exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si latencia es menor a 0.</li> <li>2. Si alguno de los parámetros String es vacío o null.</li> <li>3. Si no existe alguna de las sucursales con los códigos indicados.</li> <li>4. Si ya existe una conexión entre las dos sucursales.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

### Notas:

- Se considera que las conexiones son navegables en ambos sentidos. O sea que, si existe una conexión entre las sucursales A y B también entre B y A.
- No es necesario que todas las sucursales estén conectadas entre sí directamente, algunas sucursales están conectadas a través de otras. Por ejemplo, Si A se conecta con B y B se conecta con C, A está conectada con C a través de B.



## 12. Actualizar Conexión

Retorno **actualizarConexion**(String codigoSucursal1, String codigoSucursal2, int latencia);

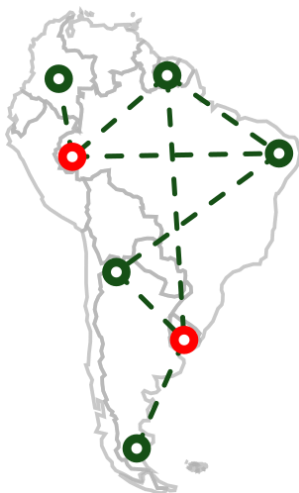
Descripción: Actualiza la latencia de la conexión entre dos sucursales (la conexión es bidireccional).

Retornos posibles	
OK	Si la latencia de la conexión fue actualizada exitosamente.
ERROR	<ol style="list-style-type: none"> <li>1. Si latencia es menor a 0.</li> <li>2. Si alguno de los parámetros String es vacío o null.</li> <li>3. Si no existe alguna de las sucursales con los códigos indicados.</li> <li>4. Si no existe una conexión entre las sucursales.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

## 13. Analizar sucursal.

Una región se compone de múltiples sucursales conectadas entre sí, y para evaluar los riesgos de conectividad, se busca determinar si una sucursal es esencial para mantener la conexión en la región. Una sucursal es considerada crítica si, al perder su conectividad, interrumpe la comunicación entre las demás sucursales de la región. Por el contrario, no se considera crítica si, al desconectarse, las demás sucursales pueden seguir funcionando sin inconvenientes.

Ejemplo:

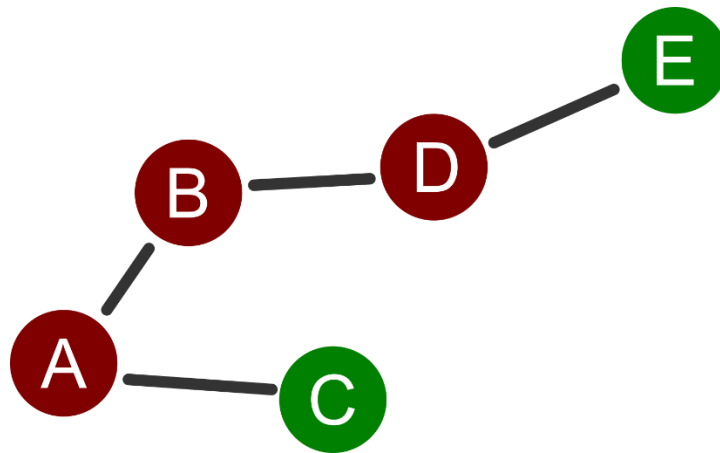


Cantidad de sucursales de la región es siete (cantidad de sucursales interconectadas en este ejemplo). Las sucursales verdes no son críticas y las sucursales rojas si lo son.

Retorno **analizarSucursal**(String codigoSucursal);

Descripción: Dado una sucursal cargar en el valorString el texto “SI” si es crítica y “NO” si no lo es.

Retornos posibles	
OK	Retorna en <i>valorString</i> la palabra SI/NO según corresponda
ERROR	1. Si el código es vacío o null. 2. Si no existe la sucursal con ese código.
NO_IMPLEMENTADA	Cuando aún no se implementó.



En el siguiente ejemplo hay tres regiones una en cada color.



## 14. Seleccionar sucursales para torneo

Retorno **sucursalesParaTorneo**(String codigoSucursalAnfitriona, int latenciaLimite);

Descripción: Retorna en valorString del retorno una lista de las sucursales ordenadas por código creciente que tengan una cantidad menor o igual a la latencia indicada como parámetro a la sucursal anfitriona.

Retornos posibles	
OK	Si el camino pudo ser calculado exitosamente. Retorna en <i>valorEntero</i> la latencia más grande de las sucursales seleccionadas. Retorna en <i>valorString</i> el listado de las sucursales ordenadas creciente por código.
ERROR	<ol style="list-style-type: none"> <li>1. Si el código de la sucursal anfitriona es vacío o null.</li> <li>2. Si no existe el código de la sucursal anfitriona.</li> <li>3. Si la latencia es menor o igual a cero.</li> </ol>
NO_IMPLEMENTADA	Cuando aún no se implementó.

Formato de retorno del *valorString*:  
codigo1;nombre1|codigo2;nombre2

### Información importante


- Se deberán **respetar los formatos de retorno** dados para las operaciones que devuelven datos.
- Está **terminantemente prohibido** el uso de clases de Java tales como **ArrayList, HashMap, etc.**
- **Ninguna** de las operaciones debe imprimir **nada** en consola.
- El sistema no debe requerir ningún tipo de interacción con el usuario por consola.
- Es obligación del estudiante mantenerse al tanto de las aclaraciones que se realicen en clase o a través del foro de aulas.
- **Se valorará la selección adecuada de las estructuras para modelar el problema y la eficiencia en cada una de las operaciones.** Deberá aplicar la metodología vista en el curso.
- El proyecto será implementado en lenguaje JAVA sobre una interfaz Sistema que se publicará en el sitio de la materia en aulas.ort.edu.uy (El uso de esta interfaz es obligatorio).
- El proyecto entregado debe compilar y ejecutar correctamente en IntelliJ IDEA.
- No se contestarán dudas sobre el obligatorio en las 48 horas previas a la entrega.
- **Defensa:** la defensa del obligatorio será en fecha y formato a coordinar, consultar con el docente. Se publicará en Aulas la fecha y horario específico con antelación. La no asistencia a la defensa implica la pérdida de todos los puntos.
- **Considere que superar las pruebas no garantiza que la operación se haya resuelto correctamente.**
- **Entrega (Se debe subir un único zip o rar con):**
  - Proyecto con las operaciones implementadas respetando las buenas prácticas y estándares vistos en el curso.
    - **La implementación debe respetar los órdenes solicitados.**
    - Incluir en la clase ImplementacionSistema los nombres y números es estudiante del equipo.
  - **Documentaciones:** Entregar un documento PDF con la información de: nombres, números de estudiante, grupo y la justificación del cumplimiento de los órdenes solicitados.

## RECORDATORIO: IMPORTANTE PARA LA ENTREGA

- **Obligatorios**

La entrega de los obligatorios será en formato digital online, a excepción de algunas materias que se entregarán en Bedelía y en ese caso recibirá información específica en el dictado de la misma.

Los principales aspectos a destacar sobre la **entrega online de obligatorios** son:

1. Ingresá al sistema de Gestión.
2. En el menú, seleccioná el ítem “Evaluaciones” y la instancia de evaluación correspondiente, que figura bajo el título “Inscripto”.
3. Para iniciar la entrega hacé clic en el ícono: 
4. Ingresá el número de estudiante de cada uno de los integrantes y hacé clic en “Agregar”. El sistema confirmará que los integrantes estén inscriptos al obligatorio y, de ser así, mostrará el nombre y la fotografía de cada uno de ellos. Una vez agregados todos los integrantes, hacé clic en “Crear equipo”.

**Cualquier integrante podrá:**

- **Modificar la integración del equipo.**
- **Subir el archivo de la entrega.**

5. Seleccioná el archivo que deseás entregar. Verificá el nombre del archivo que aparecerá en la pantalla y hacé clic en “Subir” para iniciar la entrega. Cada equipo (hasta 2 estudiantes) debe entregar **un único archivo en formato zip o rar** (los documentos de texto deben ser pdf, y deben ir dentro del zip o rar). El archivo a subir debe tener **un tamaño máximo de 40mb**

Cuando el archivo quede subido, se mostrará el nombre generado por el sistema (1), el tamaño y la fecha en que fue subido.

6. El sistema enviará un e-mail a todos los integrantes del equipo informando los detalles del archivo entregado y confirmando que la entrega fue realizada correctamente.
7. Podés cerrar la pestaña de entrega y continuar utilizando Gestión o salir del sistema.
8. La **hora tope para subir el archivo será las 21:00** del día fijado para la entrega.
9. La entrega se podrá realizar desde cualquier lugar (ej. hogar del estudiante, laboratorios de la Universidad, etc).
10. Aquellos de ustedes que presenten alguna dificultad con su inscripción o tengan inconvenientes técnicos, por favor contactarse con la Coordinadora o Coordinación adjunta antes de las 20:00hs. del día de la entrega, a través de los mails, [alamon@ort.edu.uy](mailto:alamon@ort.edu.uy) o [fernandez\\_ma@ort.edu.uy](mailto:fernandez_ma@ort.edu.uy); o telefónicamente al 29021505 - int 1156 u 1138