**Taller 2**

**Bastian Guerra Valdés**

**Daniela Mondaca Rodriguez**

**Docente Jonathan Rojas**

**Coquimbo,2022**

**Fecha: 29/05/2023**

**Introducción:**

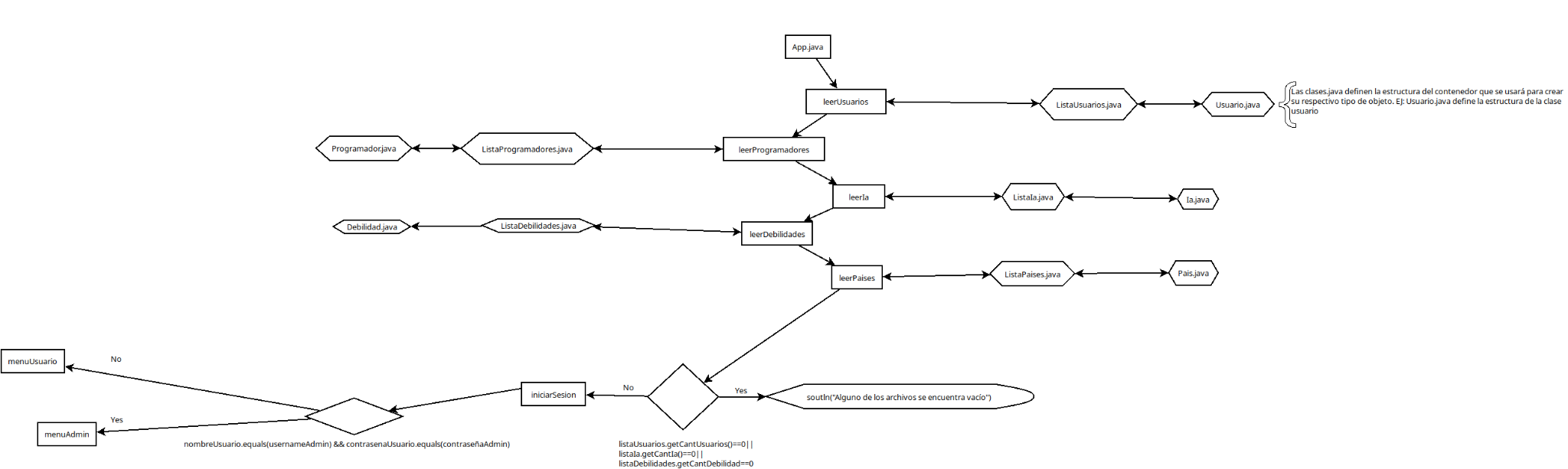
Este informe se enfoca en la realización del problema planteado en el Taller 2 de la asignatura de programación avanzada,donde en este esta enfocado a la resolución del problema en cuestión utilizando la programación orientada a objetos, uso de contenedores y lectura de los archivos correspondientes.

En este informe, proporcionaremos el diagrama del problema solicitado, así como una explicación detallada y definición de las variables, clases, objetos y funciones utilizados a lo largo del desarrollo del proyecto. Este análisis nos permitirá comprender de manera exhaustiva la estructura y funcionamiento del programa.

A lo largo de este informe, presentaremos los resultados obtenidos, los desafíos encontrados durante el proceso de desarrollo y las soluciones implementadas. Además, realizaremos un análisis crítico de las ventajas y limitaciones de la implementación propuesta, así como posibles mejoras o ampliaciones futuras.

**Desarrollo:**

**Diagrama principal**

****

**RF1: Login e início de programa:**

**RF2: Menú Usuario normal:**

**RF3: Menú Admin:**

**RF4: Guardado de datos**

**Definiciones:**

**App:  
-leerUsuarios():** Esta función lee un archivo llamado "Usuarios.txt" y guarda los usuarios en el contenedor listaUsuarios.

**-**leerProgamadores(): Esta función lee un archivo llamado "Progamadores.txt" y guarda los programadores en el contenedor listaProgamadores.

-**leerIa():** Esta función lee un archivo llamado "IA.txt" y guarda las IA en el contenedor listaIa.

-**leerDebilidades():** Esta función lee un archivo llamado "Debilidades.txt" y guarda las debilidades en el contenedor listaDebilidades.

-leerPaises(): Esta función lee un archivo llamado "Paises.txt" y guarda los países en el contenedor listaPaises.

-**iniciarSesion():** Esta función se encarga de iniciar sesión para los usuarios. Verifica si el usuario y la contraseña ingresados coinciden con los datos almacenados en el contenedor listaUsuarios y, en caso afirmativo, permite el acceso al menú de administrador.

-**menuUsuario():** Esta función representa el menú para un usuario normal después de iniciar sesión. Proporciona varias opciones, como agregar debilidades a una IA, modificar datos de usuario, modificar precisión de una IA, ver IAs, ver tipos de IA, etc.

-**menuAdmin():** Esta función representa el menú para el administrador después de iniciar sesión. Ofrece opciones como ver programadores, ver todas las IA, editar datos de programador, editar datos de IA, editar datos de usuario, crear y ver debilidades, crear IA, programador y país, dar estadísticas de países, entre otros.

-**main():** Esta función es el punto de entrada del programa. Crea los contenedores necesarios y realiza la lectura de archivos, luego llama a la función iniciarSesion para comenzar el proceso de inicio de sesión.

**Debilidad:**

-**Constructor:** El constructor se utiliza para crear objetos de la clase "Debilidad". Toma dos parámetros: "debilidad" (tipo String) y "nivelAmenaza" (tipo int). Al crear un objeto "Debilidad", el constructor asigna los valores proporcionados a las variables de instancia correspondientes.

-**Getter y Setter de "debilidad":** El método "getDebilidad()" es un getter que devuelve el valor de la variable "debilidad". El método "setDebilidad()" es un setter que se utiliza para establecer un nuevo valor para la variable "debilidad".

-**Getter y Setter de "nivelAmenaza":** El método "getNivelAmenaza()" es un getter que devuelve el valor de la variable "nivelAmenaza". El método "setNivelAmenaza()" es un setter que se utiliza para establecer un nuevo valor para la variable "nivelAmenaza".

-**toString():** Este método sobrescrito se utiliza para proporcionar una representación en forma de cadena del objeto "Debilidad". Devuelve una cadena que muestra el valor de la variable "debilidad" y el valor de la variable "nivelAmenaza" en un formato legible.

**ListaDebilidades:**

**-Constructor:**El constructor "ListaDebilidades(int max, Scanner sc)" se utiliza para crear un objeto de la clase "ListaDebilidades". Recibe dos parámetros: "max", que representa la capacidad máxima de la lista de debilidades, y "sc", que es un objeto Scanner utilizado para leer la entrada del usuario. El constructor inicializa las variables de instancia y crea un arreglo de objetos "Debilidad" con tamaño máximo "max".

-**getCantDebilidad():** El método "getCantDebilidad()" es un getter que devuelve el valor actual de la variable "cantDebilidad". Permite obtener la cantidad de debilidades actualmente almacenadas en la lista.

-**ingresarDebilidad():**Este método se utiliza para ingresar una debilidad en la lista de debilidades. Recibe como parámetro un objeto "Debilidad" y verifica si aún hay espacio disponible en la lista. Si hay espacio, agrega la debilidad al arreglo "listaDebilidades" y aumenta el contador "cantDebilidad". Retorna "true" si se pudo ingresar la debilidad, o "false" si la lista está llena.

-**CrearDebilidad():**Este método se encarga de crear una nueva debilidad. Muestra un menú de debilidades existentes y solicita al usuario si desea crear una nueva debilidad. Si la respuesta es "S" (Sí), se solicitan los datos de la nueva debilidad (nombre y nivel de amenaza) y se verifica que el nivel de amenaza esté dentro de los parámetros establecidos (1 a 5). Si los datos son válidos, se crea un objeto "Debilidad" con los valores ingresados y se llama al método "ingresarDebilidad()" de la lista de debilidades para agregar la nueva debilidad. Se muestra un mensaje indicando que se ha creado la debilidad.

-**toString():**Este método sobrescrito se utiliza para proporcionar una representación en forma de cadena de la lista de debilidades. Recorre el arreglo "listaDebilidades" y obtiene la representación en forma de cadena de cada objeto "Debilidad" presente en la lista. Retorna una cadena que muestra todas las debilidades almacenadas en la lista. Si no hay debilidades, muestra un mensaje indicando que no existen datos en debilidades.

La clase "Ia" tiene las siguientes variables miembro privadas:

"nombre": representa el nombre de la inteligencia artificial.

"lenguaje": representa el lenguaje en el que está programada la inteligencia artificial.

"nivelAmenaza": representa el nivel de amenaza asociado a la inteligencia artificial.

"debilidad": representa la debilidad de la inteligencia artificial.

"pais": representa el país de origen de la inteligencia artificial.

"precision": representa la precisión de la inteligencia artificial.

"tipo": representa el tipo de la inteligencia artificial.

"idCreador": representa el identificador del creador de la inteligencia artificial.

-**Constructor:**La clase tiene un constructor público "Ia" que se utiliza para crear objetos de la clase "Ia". El constructor recibe los valores para todas las variables miembro y los asigna a las correspondientes.

**Getters y Setters:**Para cada variable miembro, la clase proporciona un par de métodos getter y setter. Los métodos getter se utilizan para obtener el valor de una variable y los métodos setter se utilizan para establecer un nuevo valor para una variable.

**ToString:**La clase sobrescribe el método "toString()" para proporcionar una representación en forma de cadena del objeto "Ia". El método devuelve una cadena que muestra los valores de todas las variables miembro de la clase.

**ListaIa:**

**-Constructor:** Es el método que se utiliza para crear un objeto de la clase "ListaIa". Toma dos parámetros: "max" que representa el tamaño máximo de la lista y "sc" que es un objeto de la clase Scanner utilizado para leer la entrada del usuario.

**crearIa()**: Este método se utiliza para crear un objeto "Ia" y agregarlo a la lista "listaIa". Solicita al usuario ingresar varios datos, como el nombre, lenguaje, nivel de amenaza, debilidad, país, precisión y tipo de la IA. Luego, crea un objeto "Ia" con los datos ingresados y lo agrega a la lista "listaIa".

**ingresarIa():** Este método se utiliza para agregar un objeto "Ia" a la lista "listaIa". Verifica si la cantidad actual de objetos en la lista es menor que el tamaño máximo permitido. Si es así, agrega el objeto a la lista y devuelve "true". De lo contrario, devuelve "false".

**getCantIa()**: Este método devuelve la cantidad de objetos "Ia" en la lista "listaIa".

**verIaTipo():** Este método muestra por pantalla el nombre y tipo de cada objeto "Ia" en la lista "listaIa".

**verIaNombre():** Este método muestra por pantalla el nombre de cada objeto "Ia" en la lista "listaIa", ordenados alfabéticamente.

**verIaPrecision():** Este método muestra por pantalla el nombre y precisión de cada objeto "Ia" en la lista "listaIa".

**verIaPais():** Este método muestra por pantalla el nombre y país de cada objeto "Ia" en la lista "listaIa".

**verIaNivel():** Este método muestra por pantalla el nombre y nivel de amenaza de cada objeto "Ia" en la lista "listaIa".

**verIa():** Este método muestra un menú que permite al usuario seleccionar qué filtro desea aplicar a la lista "listaIa". Los filtros disponibles son: tipo, nombre (orden alfabético), precisión, país y nivel de amenaza. Dependiendo de la opción seleccionada, se llama al método correspondiente para mostrar los objetos "Ia" que cumplen con el filtro seleccionado.

**buscarId():** Este método busca un objeto "Ia" en la lista "listaIa" utilizando un identificador (id). Devuelve "true" si encuentra un objeto con el id especificado, o "false" en caso contrario.

**editarIa():** Este método permite al usuario editar los datos de un objeto "Ia" en la lista "listaIa". El usuario debe ingresar el id del objeto que desea editar y luego se muestra un menú con opciones para modificar diferentes atributos del objeto, como nombre, nivel de amenaza, debilidad, precisión, país y id. Los cambios realizados se reflejan en el objeto "Ia" correspondiente.

**porcentajePais():** Este método muestra un menú que permite al usuario seleccionar qué tipo de estadísticas de países desea ver. Las opciones son: porcentaje según países, porcentaje según ciudad o salir. Dependiendo de la opción seleccionada, se calcula y muestra el porcentaje de creación de IA y programadores en cada país o ciudad.

**Pais:**

**Constructor:** El constructor es un método especial que se utiliza para crear objetos de la clase. En este caso, el constructor de la clase "Pais" recibe tres argumentos: "pais" (nombre del país), "cantRegiones" (cantidad de regiones) y "regiones" (arreglo de cadenas que representa las regiones del país). El constructor inicializa los atributos "pais" y "regiones" con los valores proporcionados.

**Getter y Setter de "pais":** Estas funciones permiten acceder y modificar el atributo "pais" de la clase. El getter (getPais) devuelve el valor actual de "pais", mientras que el setter (setPais) permite asignar un nuevo valor a "pais".

**Getter y Setter de "regiones":** Estas funciones permiten acceder y modificar el atributo "regiones" de la clase. El getter (getRegiones) devuelve el arreglo de regiones actual, mientras que el setter (setRegiones) permite asignar un nuevo arreglo de regiones.

**toString:** Esta función sobrescribe el método toString heredado de la clase Object. Devuelve una cadena de caracteres que representa el objeto "Pais". La cadena incluye el valor de "pais" y el arreglo de "regiones" en formato de texto legible.

**ListaPaises:**

**Constructor:** El constructor es un método especial que se utiliza para crear objetos de la clase. En este caso, el constructor de la clase "ListaPaises" recibe dos argumentos: "max" (la cantidad máxima de países que puede contener la lista) y "sc" (un objeto Scanner utilizado para la entrada de datos). El constructor inicializa los atributos "max" y "sc" con los valores proporcionados, y también inicializa el arreglo "paises" con tamaño "max" y la variable "cantPaises" con valor 0.

**crearPais:** Esta función se utiliza para crear un nuevo país y agregarlo a la lista. Se realiza a través de un menú administrativo. La función solicita al usuario ingresar el nombre del país y la cantidad de regiones. Luego, se crea un nuevo objeto "Pais" con los valores proporcionados y se agrega a la lista utilizando el método "ingresarPais". Finalmente, se muestra en pantalla la representación en formato de texto del país creado.

**ingresarPais:** Esta función se utiliza para ingresar un objeto "Pais" a la lista de países. Verifica si aún hay espacio disponible en la lista (comparando "cantPaises" con "max"). Si hay espacio, agrega el país al arreglo "paises", incrementa "cantPaises" y devuelve "true". Si no hay espacio disponible, devuelve "false".

**getCantPaises:** Esta función devuelve la cantidad actual de países en la lista.

**getPais:** Esta función devuelve el nombre del país en la posición "i" del arreglo "paises".

**regiones:** Esta función devuelve el arreglo de regiones del país en la posición "i" del arreglo "paises".

**toString:** Esta función sobrescribe el método toString heredado de la clase Object. Devuelve una cadena de caracteres que representa la lista de países. Si no hay países en la lista, muestra el mensaje "No existen paises". De lo contrario, recorre el arreglo "paises" y concatena la representación en formato de texto de cada país, separados por saltos de línea.

**Progamador:**

**Constructor:** El constructor es un método especial que se utiliza para crear objetos de la clase. En este caso, el constructor de la clase "Programador" recibe varios argumentos para inicializar los atributos del programador, como el "id" (identificador), "nombre", "apellido", "aniosExperiencia" (años de experiencia), "lenguajes" (arreglo de lenguajes de programación), "cantLenguajes" (cantidad de lenguajes), "pais" y "ciudad". El constructor asigna los valores proporcionados a los atributos correspondientes.

**Getter and Setters**: Son métodos utilizados para acceder y modificar los valores de los atributos de la clase. Para cada atributo, se proporciona un par de funciones: un getter para obtener el valor del atributo y un setter para establecer un nuevo valor para el atributo. Estas funciones siguen la convención de nombres getter y setter estándar. Por ejemplo, "getId" devuelve el valor del atributo "id", mientras que "setId" establece un nuevo valor para el atributo "id".

**toString:** Esta función sobrescribe el método toString heredado de la clase Object. Devuelve una cadena de caracteres que representa al programador y sus atributos. La cadena contiene los valores de los atributos del programador, incluyendo el "id", "nombre", "apellido", "aniosExperiencia", "lenguajes" (representado como un arreglo), "pais" y "ciudad". La representación de los lenguajes utiliza el método "Arrays.toString" para convertir el arreglo de lenguajes en una cadena legible.

**ListaProgamadores:**

**Constructor:** El constructor de la clase inicializa los atributos de la lista de programadores. Recibe dos parámetros: "max" para indicar el tamaño máximo de la lista y "sc" de tipo Scanner para leer la entrada del usuario.

**buscarId():** Esta función busca un programador en la lista por su ID. Recorre la lista de programadores y compara el ID buscado con el ID de cada programador. Devuelve true si se encuentra el ID y false en caso contrario.

**ingresarProgramador():** Esta función permite ingresar un programador a la lista de programadores. Verifica si la lista aún tiene capacidad para agregar programadores y, si es así, agrega el programador a la lista y aumenta el contador de programadores.

**crearProgramador():** Esta función permite al administrador crear un nuevo programador. Solicita al usuario ingresar información como el ID, nombre, apellido, años de experiencia, lenguajes que domina, país y ciudad. Luego crea un nuevo objeto de la clase Programador y lo agrega a la lista de programadores.

**getCantProgramadores():** Esta función devuelve la cantidad de programadores que están en la lista.

**verProgramadoresPais():** Esta función muestra por pantalla los nombres y países de todos los programadores de la lista.

**verProgramadoresCiudad():** Esta función muestra por pantalla los nombres y ciudades de todos los programadores de la lista.

**verProgramadoresAniosExperiencia():** Esta función muestra por pantalla los nombres y años de experiencia de todos los programadores de la lista.

**verProgramadoresLenguajes():** Esta función muestra por pantalla los nombres y la cantidad de lenguajes que domina cada programador de la lista.

**verProgramadoresId():** Esta función muestra por pantalla los nombres y los ID de los programadores ordenados de mayor a menor.

**verProgramadores():** Esta función muestra un menú para que el administrador pueda filtrar y ver los programadores según diferentes criterios, como país, ciudad, años de experiencia, cantidad de lenguajes conocidos y ordenados por ID.

**editarProgramador():** Esta función permite al administrador editar los datos de un programador existente. Primero busca al programador por su ID y luego muestra un menú para que el administrador elija qué datos modificar, como agregar lenguaje, agregar años de experiencia, modificar país, modificar ciudad, modificar ID, modificar nombre o modificar apellido.

**buscarPais():** Esta función busca la cantidad de programadores que pertenecen a un determinado país.

**getRegion():** Esta función devuelve la ciudad del programador en la posición 'i' de la lista.

**toString():** Esta función devuelve una representación en cadena de la lista de programadores, mostrando la información de cada uno de ellos.

**Usuario:**

**Constructor:** El constructor es un método especial utilizado para inicializar los atributos de un objeto cuando se crea una instancia de la clase. En este caso, el constructor de la clase "Usuario" recibe tres parámetros: "nombre", "contraseña" e "id". Estos valores se utilizan para inicializar los atributos correspondientes del objeto.

**Getter y Setter para "nombre":** Los métodos "getNombre()" y "setNombre(String nombre)" son métodos de acceso utilizados para obtener y establecer el valor del atributo "nombre" respectivamente. El método "getNombre()" devuelve el valor actual del atributo "nombre", mientras que el método "setNombre(String nombre)" asigna un nuevo valor al atributo "nombre".

**Getter y Setter para "contraseña":** Los métodos "getContraseña()" y "setContraseña(String contraseña)" son métodos de acceso similares utilizados para obtener y establecer el valor del atributo "contraseña" respectivamente.

**Getter y Setter para "id":** Los métodos "getId()" y "setId(int id)" son métodos de acceso utilizados para obtener y establecer el valor del atributo "id" respectivamente.

**toString():** El método "toString()" es un método sobrescrito de la clase Object que devuelve una representación en forma de cadena del objeto. En este caso, el método "toString()" se ha implementado para devolver una cadena que contiene los valores de los atributos "nombre", "contraseña" e "id" del objeto "Usuario".

**ListaUsuarios:**

**Constructor:** El constructor es un método especial utilizado para inicializar los atributos de un objeto cuando se crea una instancia de la clase. En este caso, el constructor de la clase "ListaUsuarios" recibe un parámetro "max" que representa el tamaño máximo del contenedor de usuarios. El constructor inicializa el contenedor de usuarios, establece la cantidad de usuarios inicial en cero y guarda el tamaño máximo.

**Getter y Setter para "cantUsuarios":** Los métodos "getCantUsuarios()" y "setCantUsuarios(int cantUsuarios)" son métodos de acceso utilizados para obtener y establecer el valor del atributo "cantUsuarios" respectivamente. El método "getCantUsuarios()" devuelve la cantidad actual de usuarios en el contenedor.

**Getter y Setter para "listaUsuarios":** Los métodos "getListaUsuarios()" y "setListaUsuarios(Usuario[] listaUsuarios)" son métodos de acceso utilizados para obtener y establecer el valor del atributo "listaUsuarios" respectivamente. El método "getListaUsuarios()" devuelve el arreglo de usuarios actualmente almacenados en el contenedor.

**Getter y Setter para "max":** Los métodos "getMax()" y "setMax(int max)" son métodos de acceso utilizados para obtener y establecer el valor del atributo "max" respectivamente. El método "getMax()" devuelve el tamaño máximo del contenedor.

**Ingreso de usuarios en el contenedor:** La función "ingresarUsuario(Usuario usuario)" permite agregar un usuario al contenedor de usuarios. Verifica si la cantidad actual de usuarios en el contenedor es menor que el tamaño máximo. Si es así, agrega el usuario al arreglo de usuarios, incrementa la cantidad de usuarios en uno y devuelve "true". En caso contrario, devuelve "false" indicando que el contenedor está lleno y no se pudo agregar el usuario.

**Verificar usuario:** La función "verificarUsuario(String nombreUsuario, String contrasenaUsuario)" busca un usuario en el contenedor de usuarios a través de su nombre de usuario y contraseña. Recorre el arreglo de usuarios y compara el nombre de usuario y contraseña proporcionados con los del usuario actual. Si se encuentra una coincidencia, devuelve el objeto Usuario correspondiente. Si no se encuentra ninguna coincidencia, devuelve "null".

**Generar nuevo usuario:** La función "generarNuevoUsuario(String nombreUsuario)" genera un nuevo nombre de usuario utilizando el nombre proporcionado y un número aleatorio de 5 dígitos. Retorna una cadena que combina el nombre de usuario y el número aleatorio.

**Generar nueva ID:** La función "generarNuevaId()" genera un nuevo identificador de usuario aleatorio. Recorre el arreglo de usuarios y verifica si el identificador generado ya existe. Si encuentra una coincidencia, genera un nuevo número aleatorio. Retorna el identificador generado que no existe en el arreglo de usuarios.

**Crear usuario:** La función "crearUsuario(String nombreUsuario, String contrasenaUsuario)" crea un nuevo objeto Usuario utilizando el nombre de usuario, contraseña y una ID generada mediante las funciones "generarNuevoUsuario" y "generarNuevaId". Retorna el objeto Usuario creado.

**Buscar ID:** La función "buscarId(int id)" busca si un ID dado existe en el contenedor de usuarios. Recorre el arreglo de usuarios y compara el ID proporcionado con los IDs de los usuarios almacenados. Si encuentra una coincidencia, devuelve "true". Si no encuentra ninguna coincidencia, devuelve "false".

**Editar usuario:** La función "editarUsuario()" permite editar los datos de un usuario en el contenedor. Solicita ingresar un ID y verifica si existe en el contenedor. Si el ID existe, muestra un menú para editar los datos del usuario, como el nombre, la contraseña y el ID. Realiza las modificaciones correspondientes en el objeto Usuario y muestra los datos actualizados.

**toString():** El método "toString()" es un método sobrescrito de la clase Object que devuelve una representación en forma de cadena del objeto. En este caso, el método "toString()" se ha implementado para mostrar los datos de todos los usuarios almacenados en el contenedor, devolviendo una cadena con la información de cada usuario en una línea separada. Si el contenedor está vacío, muestra un mensaje indicando que no existen usuarios.