



**MÉTODO DE DISEÑO DE LA INGENIERÍA
COMPUTACIÓN Y ESTRUCTURAS DISCRETAS I
TAREA INTEGRADORA II**

By:

Juan David Bahamon Rodriguez

A00375826

Carlos Javier Bolaños Riascos

A00377995

Samuel Hernandez Espitia

A00375392

David Esteban Peñaranda Scarpetta

A00375827

FASE 1: IDENTIFICACIÓN DEL PROBLEMA

El equipo VIP de simulación de la Facultad de Ingeniería de la Universidad Icesi, ha contactado a nuestro equipo de desarrolladores en formación para ser parte de su subproyecto de investigación. Nuestra tarea asignada es el diseño e implementación de un prototipo de software que permita la gestión eficiente de operaciones CRUD (Create, Read, Update and Delete) sobre una base de datos de personas de nuestro continente. Los datos a generar son: código (que es autogenerado por nuestro proyecto de software), nombre, apellido, sexo, nacionalidad y fotografía.

La cantidad de datos a gestionar en nuestro proyecto es 1 millón, por lo que debemos construir un modelo eficiente, relacionando las buenas prácticas de la ingeniería de software y las estructuras de datos que mejor se ajusten a la necesidad. Además, se nos ha pedido buscar una solución para permitir la interacción intuitiva y cómoda entre el software y el usuario (Implementación de una interfaz gráfica).

Las condiciones entregadas por el equipo VIP de Simulación son:

1. El registro de un millón de personas debe ser generado con dos bases de datos que han sido puestas a nuestra disposición, que contienen nombres y apellidos, los cuales deben ser combinados todos los nombres con todos los apellidos.
2. Se debe de generar de forma aleatoria la fecha de nacimiento de cada una de las personas que nacen en el continente americano.
3. Teniendo en cuenta una las estaturas promedio de los hombres y de las mujeres, se debe de generar aleatoriamente la estatura de cada una de las personas según su género y que esta tenga sentido.
4. Cada persona al nacer tiene una nacionalidad, por lo que debe ser generada teniendo en cuenta los porcentajes relativos de la población de cada país con respecto al continente americano, en caso de que las nacionalidades generadas sean diferentes a la población total, se debe de realizar una distribución del sobrante o faltante de tal forma que quede la misma población total.
5. La fotografía de la persona debe ser generada aleatoriamente en un sitio web, teniendo en cuenta que no importa el género, edad, estatura y otros.

En cuanto a la funcionalidad que ofrece el programa al usuario son las siguientes:

1. El programa debe contener una barra de progreso, el cual en caso de que la operación se demore más de 1 segundo debe de indicar el tiempo que se demoró en finalizar.
2. El usuario tiene la posibilidad de elegir cuántos registros desea generar, para ello se dispone de un campo de texto el cual permite digitar el número de registros, teniendo en cuenta que hay un valor máximo.
3. El programa debe contar con una opción para guardar los datos generados en la base de datos, permitiendo realizar futuramente una consulta.
4. La interfaz es amigable y brinda las siguientes funcionalidades al usuario:
 - Dispone de campos de texto los cuales permiten al usuario agregar una nueva persona a la base de datos.
 - Permite que el usuario realice la búsqueda de una persona, seleccionando un solo criterio, bien sea nombre, apellido, nombre completo o código.
 - Dispone de campos de texto que permiten actualizar datos de una persona existente.
 - Permite seleccionar una persona y eliminarla de la base de datos.
 - Para la funcionalidad de búsqueda con los criterios de nombre, apellido y nombre completo, a medida que vaya el usuario ingresando las letras, cuando coincidan menos de 20 elementos, debe de aparecer una lista emergente con los nombres de la base de datos que coinciden.

Análisis de requerimientos:

Cliente	Equipo VIP de simulación de la Universidad Icesi.
Usuario	Integrantes del Equipo VIP de simulación de la Universidad Icesi.
Contexto del problema	<p>El equipo VIP de simulación de nuestra universidad, nos ha contactado para asignarnos la tarea de hacer un prototipo de software que haga gestión de un dataset con Operaciones CRUD. Nuestro proyecto debe tener la capacidad de gestionar 1 millón de datos.</p> <p>Para cumplir con nuestra tarea, debemos conocer estadísticas que nos ayuden a</p>

	<p>generar datos en rangos coherentes. Al ser una simulación de datos planteada para el continente americano, debemos usar nacionalidades que pertenezcan a América. Por otra parte, necesitamos datos base (nombre y apellidos) para iniciar el proceso de creación y simulación.</p>
Requerimientos funcionales	<p>GEDT1: Importar de los datasets suministrados.</p> <p>GEDT2: Generar datos.</p> <p>UIT1: Permitir generación de datos.</p> <p>GEDT3: Guardar datos generados.</p> <p>ALDT1: Almacenar los datos (Persistencia).</p> <p>UITI2: Permitir creación de datos.</p> <p>UITI3: Permitir búsqueda de datos.</p> <p>UITI4: Permitir actualización/edición de datos.</p> <p>UITI5: Permitir eliminación de datos.</p> <p>GEDT4: Crear datos.</p> <p>GEDT5: Consultar datos.</p> <p>GEDT6: Actualizar/Editar datos</p> <p>GEDT7: Eliminar datos.</p>
Requerimientos no funcionales	<p>El prototipo de software debe estar construido e implementado en el lenguaje de programación Java. Además de contener una interfaz gráfica con elementos de Javafx y pruebas unitarias.</p> <p>El manejo de los datos, deben estar ordenados y balanceados siguiendo la teoría de los Arboles ABB y AVL.</p>

Identificador	GEDT1: Importar de los datasets suministrados		
Resumen	Dados los datasets de nombres, apellidos, nacionalidad, distribución de edad y estatura el sistema debe leerlos para que más adelante poder generar y combinar los datos a petición del usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre chicos	File (csv)	
	nombre chicas	File (csv)	
	apellidos	File (csv)	
	distribución edad	File (csv)	
	distribución estatura	File (csv)	
	nacionalidad	File (csv)	
Actividades generales necesarias para obtener resultados	Se leen los datasets.		
Resultado o postcondición	Lectura de los datasets.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección

Identificador	GEDT2: Generar datos.		
Resumen	El programa debe de tomar los datasets de nombres y apellidos, edades, estaturas y nacionalidad, para poder hacer uso de ellos y generar cada una de las personas en la base de datos.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre chicos	File (csv)	
	nombre chicas	File (csv)	
	apellidos	File (csv)	
	edad	int	
	estatura	int	
	género	String	
	nacionalidad	String	
Actividades generales necesarias para obtener resultados	<ol style="list-style-type: none"> 1. El programa debe de tomar los datasets de los nombres y apellidos, los cuales deben ser combinados. 2. Tomar los datasets de la distribución de las edades, estaturas y nacionalidad para generar los datos necesarios para crear una persona. 		
Resultado o postcondición	Los datos de cada una de las personas ha sido generado exitosamente.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	nombre	String	
	apellidos	String	
	edad	int	
	estatura	int	
	género	String	
	nacionalidad	String	

Identificador	UIT1: Permitir generación de datos.		
Resumen	La interfaz del prototipo debe tener un módulo que permita iniciar el proceso de generación de datos. Para iniciar la generación, el usuario ingresa la cantidad de registros a generar. Además, se debe mostrar una barra de proceso que indique cómo va la solicitud realizada por el usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	cantidad de registros a generar	int	
Actividades generales necesarias para obtener resultados	<ol style="list-style-type: none"> 1. Leer cantidad de registros 2. Iniciar proceso de generación 3. Mostrar barra de proceso 		
Resultado o postcondición	Datos/Registros generados		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	Registro	Person	Es de tipo Person, ya que al generar los datos, en el backend lo que se hace es crear objetos de tipo Person con estos datos.

Identificador	GEDT3: Guardar datos generados.		
Resumen	El programa debe de permitir tomar cada uno de los datos generados y guardarlos en una base de datos.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	Registro	Person	Se guarda el objeto de tipo Person. Ahí están los datos generados por el sistema que deseamos guardar.
Actividades generales necesarias para obtener resultados	Cuando se realiza la generación de los datos de la persona, el sistema debe de tomar esos datos y guardarlos en una base de datos.		
Resultado o postcondición	Los datos han sido guardados exitosamente.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección

Identificador	ALDT1: Almacenar los datos (Persistencia).		
Resumen	El sistema debe de tomar todos los datos que han sido generados y cuando el usuario pulse el botón de guardar o se cierre la pestaña de la interfaz, se deben de guardar cada uno de los datos, de tal forma que en el futuro cuando se abra el programa, estos datos sigan a disposición del usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre	String	El campo ha sido generado
	apellidos	String	El campo ha sido ingresado
	edad	int	El campo ha sido generado
	estatura	int	El campo ha sido generado
	género	String	El campo ha sido generado
	nacionalidad	String	El campo ha sido generado
Actividades generales necesarias para obtener resultados	<ol style="list-style-type: none"> 1. Al pulsar el botón de guardado, el sistema guarda todos los datos (persistencia). 2. Cuando el usuario vuelva a abrir el programa, todos los datos que han sido previamente guardados, permanecerán en el programa. 		
Resultado o postcondición	Los datos han sido guardados exitosamente y permanecen en el programa para un futuro uso.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección

Identificador	UITI2: Permitir creación de datos.		
Resumen	El sistema debe permitir al usuario ingresar al módulo de crear. Ahí, el usuario ingresará los datos necesarios en los campos solicitados para la creación de una persona en la base de datos.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre	String	
	apellidos	String	
	edad	int	
	estatura	int	Se deben ingresar en centímetros
	nacionalidad	String	
	género	String	
Actividades generales necesarias para obtener resultados	1. Leer los datos ingresados en los campos. 2. Pasar los datos como parámetros a los métodos para la creación del objeto persona. (Ver en GEDT4)		
Resultado o postcondición	Registro creado.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	Notificación “Registro completado”	Alerta	

Identificador	UITI3: Permitir búsqueda de datos.		
Resumen	La interfaz contiene una opción para seleccionar el criterio deseado, y posteriormente un campo de texto el cual permite al usuario realizar la búsqueda. A medida que el usuario va ingresando el texto, cuando haya menos de 20 coincidencias, se despliega una lista con recomendaciones de los usuarios.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	criterio	acción event	
	campoTexto	String	El criterio ha sido seleccionado
Actividades generales necesarias para obtener resultados	<ol style="list-style-type: none"> 1. La interfaz brinda un espacio de selección del criterio. 2. La interfaz brinda un campo de texto. 3. La interfaz brinda un campo de visualización de la lista de recomendaciones. 4. La interfaz brinda un espacio para visualizar la persona que han sido buscada 		
Resultado o postcondición	La búsqueda de la persona ha sido exitoso o fracasado		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	person	Person	

Identificador	UITI4: Permitir actualización/edición de datos.		
Resumen	El sistema debe permitir al usuario ingresar al módulo de edición para actualizar los datos de una persona. Para eso, el usuario debe seleccionar el registro a editar, para poder sobrescribir los datos que este vea necesarios.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre	String	
	apellidos	String	
	edad	int	
	estatura	int	se debe ingresar en centímetros
	género	String	
	nacionalidad	String	
Actividades generales necesarias para obtener resultados	<ol style="list-style-type: none"> 1. Leer los cambios realizados 2. Pasar los datos cómo parámetros a los métodos para la actualización/edición del objeto persona. (Ver en GEDT6) 		
Resultado o postcondición	Registro actualizado.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	Notificación “Actualización realizada con éxito”	Alert	

Identificador	UITI5: Permitir eliminación de datos.		
Resumen	El sistema debe permitir al usuario ingresar el módulo de eliminar el registro de una persona. Para eso, el usuario selecciona un usuario y da inicio al proceso al seleccionar el botón “Eliminar registro”.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	registro	Person	
Actividades generales necesarias para obtener resultados	1. Seleccionar el registro a eliminar. 2. Pasar el registro como parámetro a los métodos para la eliminación del objeto persona. (Ver en GEDT7)		
Resultado o postcondición	Registro eliminado.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	Notificación “Registro eliminado con éxito”	Alert	

Identificador	GEDT4: Crear datos.		
Resumen	Iniciado el proceso de creación de registro desde el módulo de creación (Interfaz gráfica : UITI2). El sistema debe crear un registro. El id de cada persona, es autogenerado por el sistema, este es importante para que se pueda finalizar la creación.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre	String	
	apellidos	String	
	edad	int	
	estatura	int	
	nacionalidad	String	
	género	String	
Actividades generales necesarias para obtener resultados	1. Se crea en nuevo registro 2. Se guarda en la base de datos.		
Resultado o postcondición	Registro creado.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	insert	boolean	True = Se agregó False = No se agregó.

Identificador	GEDT5: Consultar datos.		
Resumen	Iniciado el proceso de búsqueda de un registro (guardado con anterioridad) desde el módulo de consulta (Interfaz gráfica : UITI3), el sistema debe buscar en su base de datos los registros similares que encajen con el criterio de búsqueda seleccionado por el usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	criterio de búsqueda	String	
Actividades generales necesarias para obtener resultados	1. Buscar registros similares 2. Conexión con interfaz gráfica 3. Mostrar aproximaciones de la búsqueda.		
Resultado o postcondición	Registro seleccionado. (Se busca entre las posibles y se selecciona).		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	Registro	Person	

Identificador	GEDT6: Actualizar/Editar datos		
Resumen	Iniciado el proceso de actualización de un registro (guardado con anterioridad) en el módulo de edición (Interfaz gráfica : UITI4). El sistema debe guardar los cambios realizados al registro seleccionado por el usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	nombre	String	
	apellidos	String	
	edad	int	
	estatura	int	se debe ingresar en centímetros
	género	String	
	nacionalidad	String	
Actividades generales necesarias para obtener resultados	1. Guardar los cambios.		
Resultado o postcondición	Registro actualizado		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	actualización	boolean	False = No se actualizó. True = Se actualizó.

Identificador	GEDT7: Eliminar datos.		
Resumen	Iniciado el proceso de eliminación de registro (previamente guardado) en el módulo eliminar. El sistema debe eliminar el registro seleccionado por el usuario.		
Entradas	Nombre de la entrada	Tipo de dato	Condición de repetición o selección
	registro	Person	
Actividades generales necesarias para obtener resultados	1. Eliminar registro.		
Resultado o postcondición	Registro eliminado.		
Salidas	Nombre de la salida	Tipo de dato	Condición de repetición o selección
	delete	boolean	False = Si no se eliminó. True = Si se eliminó

FASE 2: RECOPIACIÓN DE INFORMACIÓN NECESARIA.

- **ComboBox:** Se usa para mostrar datos en un cuadro combinado desplegable. De forma predeterminada, el ComboBox control aparece en dos partes: la parte superior es un cuadro de texto que permite al usuario escribir un elemento de lista. La segunda parte es un cuadro de lista que muestra una lista de elementos desde los que el usuario puede seleccionar uno. **Recuperado de:** <https://docs.microsoft.com/es-es/dotnet/desktop/winforms/controls/combobox-control-overview-windows-forms?view=netframeworkdesktop-4.8>
- **Librería SwingX:** Esta es una librería para agregar componentes gráficos de java. **Obtener librería e importar en el proyecto en:** <https://elblogdecodigo.wordpress.com/2014/12/26/swingx-una-libreria-de-componentes-graficos-para-java/>
- **Implementación de auto complete en java:** <https://stackoverflow.com/questions/14849176/implementing-auto-complete-in-java-am-i-doing-it-right>
- **Class AutoCompleteDecorator,** Esta clase contiene sólo métodos de utilidad estáticos que se pueden usar para configurar la finalización automática de algunos componentes de Swing. **en:** <https://javadoc.io/static/org.swinglabs/swingx/1.6.1/org/jdesktop/swingx/autocomplete/AutoCompleteDecorator.html>

FASE 3: BÚSQUEDA DE SOLUCIONES CREATIVAS

Las alternativas que se exponen a continuación, están pensadas para buscar solución a la barra de búsqueda con sugerencias, una función muy importante para el prototipo a desarrollar. Las alternativas son:

Alternativa 1:

Primero que todo se debe crear un ComboBox, es decir un campo que es desplegable. En sus configuraciones, se permite que sea editable, es decir que el usuario pueda escribir dentro).

```
comboBox = new JComboBox();
```

```
comboBox.setEditable(true);
```

Para lograr desplegar una lista con sugerencias de búsqueda cuando el usuario digita el texto, se utiliza una librería externa llamada SwingX.

`AutoCompleteDecorator.decorate(comboBox);`

Por lo tanto cuando el usuario pulsa una tecla dentro del comboBox, se va a añadir una lista desplegable con sugerencias.

Alternativa 2:

Utilizar GlazedList, una clase personalizada llamada Autosuggestor, el cual es un ArrayList de palabras para comparar las palabras que han sido escritas por el usuario, pasando JTextField de referencia a DocumentListener, realizando una verificación, y se encarga de mostrar o no sugerencias, y de ser así qué sugerencias mostrar.

Alternativa 3:

Utilizar ObservableList, elemento usado en las interfaces con JavaFx. Una ObservableList permite ver elementos en pantalla mediante una TableView, que a su vez está compuesta de TableColumn. Lo que se plantea hacer es que al momento de que el usuario ingrese un campo de texto (una cadena String), se haga la búsqueda en la base de datos de las personas que tienen están relacionadas con el elemento y las muestre.

FASE 4: TRANSICIÓN DE LA FORMULACIÓN DE IDEAS A LOS DISEÑOS PRELIMINARES.

Con el fin de tener un mejor desarrollo del prototipo, hemos decidido complementar cada una de las alternativas propuestas en la fase anterior.

Alternativa 1:

SwingX admite auto completar y puede ser más fácil de usar que GlazedList, pues todo lo que se escribe en SwingX es `AutoCompleteDecorator.decorate(comboBox);` Y para usar esta forma, simplemente se debe de descargar un archivo jar y agregarlo a la carpeta de la biblioteca del proyecto.

Algunas de sus ventajas son:

- El diseño en Java puro posee menos limitaciones de plataforma.
- El desarrollo de componentes Swing es más activo.
- Los componentes de Swing soportan más características.

Alternativa 2:

Para este punto, debemos tener en cuenta que el número de datos a almacenar es muy grande, por lo que debemos pasar los elementos filtrados y sean parecidos a los necesarios a un tipo de lista (Puede ser arraylist o entre otras implementaciones de Java) para poder presentar los elementos en pantalla y cumplir la funcionalidad requerida.

Alternativa 3:

Para poder utilizar esta alternativa hemos decidido que se podría utilizar ObservableList acompañado de nuestro árbol AVL, la lista se utilizará para mostrar los datos encontrados con coincidencias en el árbol según sea el criterio de búsqueda seleccionado por el usuario.

FASE 5: EVALUACIÓN Y SELECCIÓN DE LA MEJOR SOLUCIÓN.

Teniendo en cuenta las diferentes posibles soluciones, se ha realizado un análisis de la solución más factible, con base en los siguientes criterios:

Criterio A. Velocidad: La alternativa es rápida en el proceso de mostrar las sugerencias que encajan con el criterio de búsqueda.

Criterio B. Compatibilidad: La alternativa es compatible con el diseño de la interfaz gráfica y con el entorno de desarrollo utilizado por el equipo de desarrolladores.

Criterio C. Funcionalidad: La alternativa es eficiente, eficaz y es aproximada a la solución deseada.

Se evaluará cada una de las alternativas asignando una calificación del 0 al 10, siendo 10 la mejor calificación y 0 la peor.

Criterio/ Alternativa	Criterio A	Criterio B	Criterio C	Total
Alternativa 1	10	10	6	26
Alternativa 2	10	8	10	28
Alternativa 3	10	10	10	30

Según la evaluación de cada una de las alternativas, con un total de 30 puntos, la mejor solución es usar **ObservableList**, puesto que esta presenta una mayor compatibilidad con los diferentes componentes de la interfaz, además su implementación es mucho más fácil y eficaz.

FASE 6: PREPARACIÓN DE INFORMES Y ESPECIFICACIONES.

Especificación del problema:

Problema: Gestionar eficientemente las operaciones CRUD sobre una base de datos de personas.

Flujo del sistema: El usuario ingresa al sistema y la interfaz le ofrece 3 opciones: crear personas, buscar y eliminar.

- Cuando el usuario desea generar datos de personas, debe de digitar un número entero, una vez se ha ingresado, el usuario pulsa el botón de generar, por lo que el sistema internamente toma ese número y genera cada una de las personas deseadas.
- Cuando el usuario desea buscar a una persona, primero que todo debe de seleccionar el criterio de búsqueda, y luego escribe el dato correspondiente a la persona que desea encontrar.
- Cuando el usuario desea eliminar a una persona, la interfaz va a mostrar cada una de las personas generadas, por lo que el usuario debe digitar el nombre y esta lista se actualizará con datos similares o recomendados, una vez se muestran los datos que se quieren eliminar, este debe seleccionarlo y pulsar el botón de eliminar.

Por ejemplo:

El usuario entra al sistema y desea generar 10 personas, por lo cual debe de ingresar a la interfaz de generación de personas, luego debe de ingresar el número 10 en el campo disponible y pulsar el botón de generar, por lo que el sistema realiza la generación de los 10 datos. Luego desea buscar a una persona, por lo que ingresa a la interfaz de búsqueda y selecciona por ejemplo el criterio de nombre y apellido, y posteriormente ingresa el nombre y el apellido de la persona en el campo disponible, por lo que el sistema busca a la persona y en caso de encontrarla la muestra, de lo contrario no se mostrará los datos de esa persona.

En el caso de eliminar se ingresa a dicha interfaz, la cual contiene un listado de personas, y está la opción de seleccionar directamente la persona y luego pulsando el botón de eliminar, una vez la persona ha sido eliminada, el listado y la base de datos se actualiza quedando con un total de 9 personas.

Hay que tener en cuenta que generar personas se puede hacer cuantas veces se requiera siempre y cuando no exceda con el límite máximo. En la búsqueda de personas siempre se

debe seleccionar un criterio y se mostrará siempre y cuando este exista. Y por el lado de la eliminación de personas, se podrá eliminar cuántas veces se requiera siempre y cuando exista por lo menos 1 persona, es decir no se podrá eliminar en caso de que hayan 0 generados.

Consideraciones y observaciones:

- Durante el proceso de desarrollo, observamos que al realizar la generación del millón de datos solicitados, hay tiempo estimado de 12 minutos, cantidad muy lejana a la condición establecida de 30 segundos, debido a su crecimiento lineal. Por lo que procedimos a mejorar y reestructurar el árbol AVL para encontrar la cantidad ideal que se ajustaba al tiempo. Evaluando la generación en un tiempo de 30 seg, se obtiene una cantidad total de 20.000 personas.
- Como ayuda al equipo de desarrolladores, los integrantes del equipo VIP de simulación nos han extendido el plazo de entrega del prototipo para el día 8 de mayo de 2022 por motivos de claridad en las necesidades y problemas en las soluciones a implementar el prototipo ideal.
- Centrándonos en el tema de visualización de imágenes, la imagen es un dato que no puede ser modificado. Cuando el programa inicia no hay existencia de imágenes, por lo que al momento de pulsar el botón de “*ver detalles*” por primera vez, se autogenera una imagen para la persona seleccionada y se guarda en una carpeta. Asimismo, cuando se selecciona la opción por segunda vez, se carga la imagen correspondiente. Finalmente, al sobrescribir los datos, se eliminan todas las imágenes que han sido guardadas durante la ejecución.
- Por otra parte, en el proceso de desarrollo encontramos limitaciones con la serialización, al ser una gran cantidad de datos que cuentan con recursividad se hacen muchos llamados, por lo que se satura y se produce una *StackOverflowException*. Para esta situación, hicimos muchos intentos, hasta que llegamos a una cantidad posible de 600 datos, por lo que se ha establecido un límite de 500 datos para evitar problemas en el proceso de serialización y deserialización en el sistema.

FASE 7: IMPLEMENTACIÓN DEL DISEÑO.

La implementación del diseño se desarrolla en el lenguaje de programación: Java. Haciendo uso de árboles AVL.

Para el prototipo a desarrollar, debemos implementar las siguientes tareas:

1. Generar la cantidad de datos ingresada.
2. Permitir buscar personas por el criterio de búsqueda seleccionado por el usuario.
3. Permitir crear personas al usuario.
4. Permitir eliminar personas cuando el usuario lo desee.
5. Permitir editar personas.

ANEXOS

Diseño caso de prueba:

Configuración de los escenarios.

Nombre	Clase	Escenario
setupStage1	AVLTest	Un objeto de la clase AVLTree. Se le agregan 5 personas. ("id", "Pedro", "lastname", 10, 0, 120, "nationality"); ("id", "Ramiro", "lastname", 10, 0, 120, "nationality"); ("id", "Omar", "lastname", 10, 0, 120, "nationality"); ("id", "Martha", "lastname", 10, 0, 120, "nationality"); ("id", "Nadia", "lastname", 10, 0, 120, "nationality");
setupStage2	AVLTest	Un objeto de la clase AVLTree. Se le agrega una persona. ("id", "Pedro", "lastname", 10, 0, 120, "nationality");
setupStage3	AVLTest	Un objeto de clase AVL Tree. Se le agrega una persona, pero esta es eliminada, pero para eso es previamente buscada con el parámetro "id". ("id", "Pedro", "lastname", 10, 0, 120, "nationality");

Objetivo de la prueba:

El objetivo de esta prueba es verificar el funcionamiento de los métodos de la clase AVLTree. En donde se realiza la inserción de las personas en el árbol AVL. Se examinarán los métodos add, delete, searchId, isEmpty, balance, checkFb, etc.

Clase	Método	Escenario	Valores de la entrada	Resultado
-------	--------	-----------	-----------------------	-----------

AVLTree	testBalance	setupStage1	N/A (Los valores a examinar están en el setupStage1)	Obtenemos un árbol balanceado, con sus nodos en la ubicación correcta según el criterio de nombre. Por lo que el método balance funciona.
AVLTree	testSearch	setupStage2	N/A (Los valores a examinar están en el setupStage2)	Se encuentra a “Pedro” en el objeto de la clase AVLTree. Por lo que el método search funciona.
AVLTree	testAdd	setupStage2	N/A (Los valores a examinar están en el setupStage2)	Se obtiene un boolean false, ya que el árbol AVL no se encuentra vacío, entre sus elementos está “Pedro”. Por lo que el método add funciona.
AVLTree	testDelete	setupStage3	N/A (Los valores a examinar están en el setupStage3)	Al comprobar el estado del árbol, vemos que este está vacío, esto dado que se usa el método delete después de usar add en el setupStage3. Por lo que el método delete funciona.

