Gestión Eficiente de Base de Datos de Personas

Danna Alexandra Espinosa Arenas

Cristian Felipe Perafan Chilito

Facultad de Ingeniería y Diseño, Universidad Icesi

09781 : Computación y estructuras discretas I.

Ph. Uram Anibal Sosa Aguirre.

2 may 2022

ÍNDICE:

FUNCTIONAL REQUIREMENTS

Requirement #1:

Nombre o Identificador	R1. Creation of people		
Resumen	The system must allow the automatic creation of persons with their respective data, for which the databases suggested by the teacher must be implemented (code (auto-generated), name, surname, sex, date of birth, height, nationality and photograph).		
Entradas	Nombre	Tipo	Condición
	Name	String	
	Surname	String	data entered above must be different from null
	Sex	String	data entered above must be different from null
	Date of birth	Date	data entered above must be different from null
	Height	Double	data entered above must be different from null
	Nationality	String	data entered above must be different from null
	Photograph	Png	data entered above must be different from null
Resultado	A person is created with the above data and a unique code is generated to identify him/her.		
Salidas	Persons		

Requirement #2:

Nombre o Identificador	R2. Persistent		
Resumen	The program must allow basic persistent storage functions (Create, Read, Update and Delete) with the system data, especially the data of persons.		
Entradas	Nombre	Tipo	Condición
	Option	String	
Resultado	The requested action is executed, e.g., read databases.		
Salidas			

Requirement #3:

Nombre o Identificador	R3. Updating data person		
Resumen	The program should allow updating a person's data.		
Entradas	Nombre	Tipo	Condición
	Name	String	The previous name of the person must be different from null
	Surname	String	The previous Surname of the person must be different from null
	Sex	String	The previous sex of the person must be different from null
	Date of birth	Date	The previous date of birth of the person must be different from null
	Height	Double	The previous height of the person must be different from null
	Nationality	String	The previous nationality of the person must be different from null
	Photograph	Png	The previous photograph of the person must be different from null
Resultado	The person's data is maintained		
Salidas			

Requirement #4:

Nombre o Identificador	R4.Deleting data person		
Resumen	The program should allow deleting a person's data.		
Entradas	Nombre	Tipo	Condición
	Code	String	This person must exist in order to be eliminated
Resultado	The person is removed from the application		
Salidas			

Requirement #5:

Nombre o Identificador	R5. Add Person		
Resumen	The program must allow to create a person with their respective data.		
Entradas	Nombre	Tipo	Condición
	Name	String	The previous name of the person must be different from null
	Surname	String	The previous Surname of the person must be different from null
	Sex	String	The previous sex of the person must be different from null
	Date of birth	Date	The previous date of birth of the person must be different from null
	Height	Double	The previous height of the person must be different from null
	Nationality	String	The previous nationality of the person must be different from null
	Photograph	Png	The previous photograph of the person must be different from null
Resultado	The person is created.		
Salidas			

Requirement #6:

Nombre o Identificador	R6 Searching by First Name		
Resumen	The program must allow a person to be searched by first name and display the list of found persons		
Entradas	Nombre	Tipo	Condición
	First Name		There must have been at least one person in the application.
Resultado	The list of search results by first name is displayed.		
Salidas	list of persons found	ArrayList	

Requirement #7:

Nombre o Identificador	R7 Searching by Last Name		
Resumen	The program must allow a person to be searched by last name and display the list of found persons		
Entradas	Nombre	Tipo	Condición
	Last name	String	There must have been at least one person in the application.
Resultado	The list of search results by last name is displayed.		
Salidas	list of person found	ArrayList	

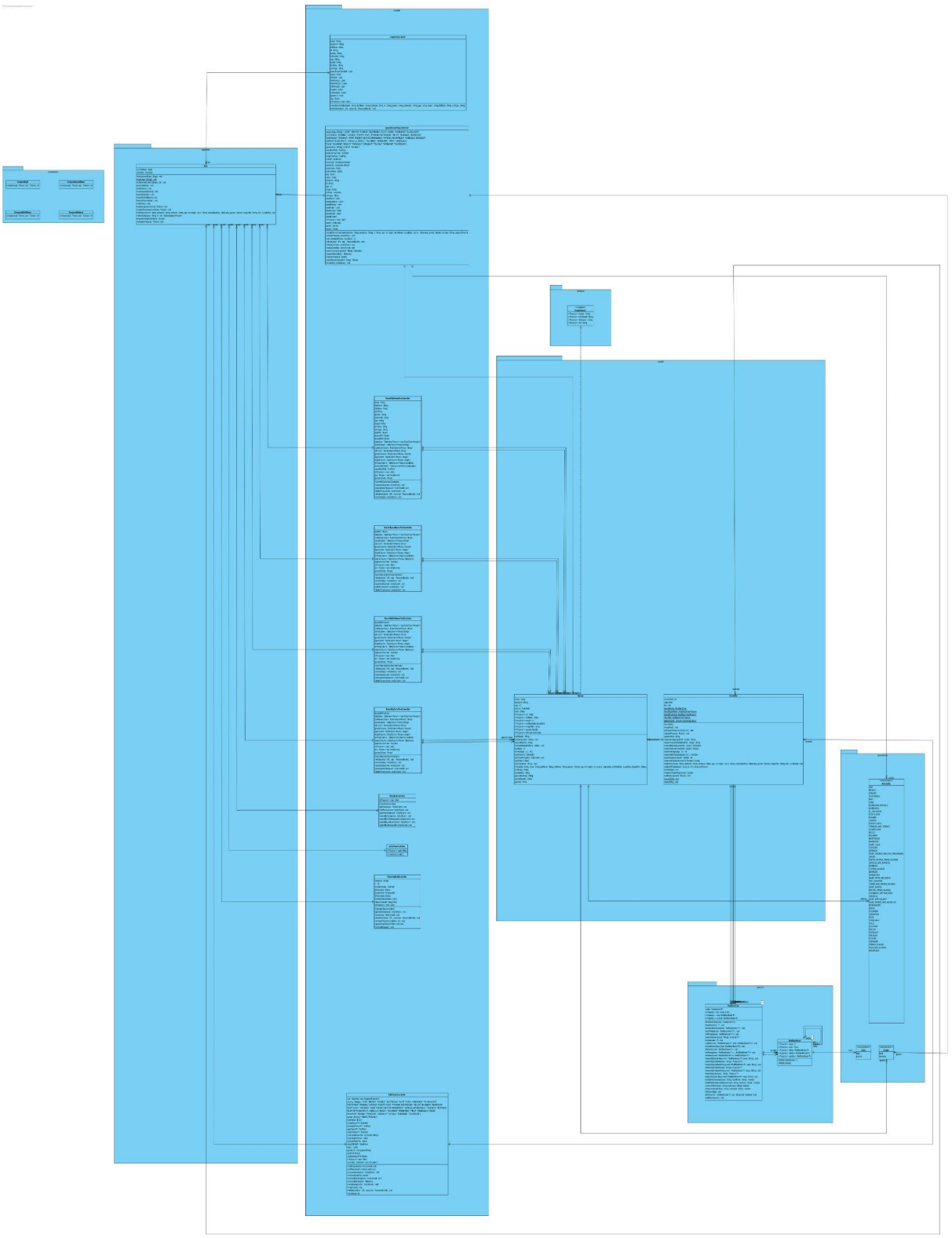
Requirement #8:

Nombre o Identificador	R7.Searching by Full Name		
Resumen	The program must allow a person to be searched by full name and display the list of found persons		
Entradas	Nombre	Tipo	Condición
	Full name	String	There must have been at least one person in the application.
Resultado	The list of search results by full name is displayed.		
Salidas	list of persons found	ArrayList	

Requirement #9:

Nombre o Identificador	R8. Searching by Code		
Resumen	The program must allow a person to be searched by code and display the list of found persons		
Entradas	Nombre	Tipo	Condición
	Code	String	There must have been at least one person in the application.
Resultado	The list of search results by code is displayed.		
Salidas	List of persons found	ArrayList	

CLASS DIAGRAM



TEST CASE DESIGN

Configuración de los Escenarios

Nombre	Clase	Escenario
setupStage1	Controller	Instancia de Person está null
setupStage2	Controller	Instancia de treeRBName, treeRBLastName,treeRBFullName,treeRBId está null.
setupStage3	Controller	ArrayList<Person> personDataA está vacío
setupStage4	Controller	ArrayList<Person> personDataA tiene dos instancias de la clase Persona
setupStage5	Controller	Instancia de randomN está null
setupStage6	Controller	Instancia de relativePathName está null
setupStage7	Controller	Instancia de relativePathLastName está null
setupStage8	Controller	randomN tiene un valor aleatorio asignado.
setupStage9	RedBlackTree	Instancia de RedBlackTree tiene n-elementos

Diseño de Casos de Prueba

Objetivo de la Prueba: Validar la correcta creación de un contacto.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Controller	addPerson	setupStage1	name=”Hugo” lastName=”Boss” fullName=”Hugo Boss” age=18 height=170 id=self-generate nationality=Colombia gender=man imagePath=urlRandom. doB=18-05-2004	True. Se ha creado una nueva persona exitosamente. Cada uno de los atributos del nuevo contacto tiene asignada correctamente la información pasada por parámetro.

Objetivo de la Prueba: Validar la correcta creación de n-Personas.

Clase	Método	Escenario	Valores de Entrada	Resultado
Controller	toCreatePerson	setupStage2	numDates=10	true. Se han creado 10 personas nuevas con atributos aleatorios.

Objetivo de la Prueba: Verificar que se haya eliminado a una persona del árbol				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Controller	deleteAPerson	setupStage4	Person person	Se elimina a una persona de la base de datos y del arbol

Objetivo de la Prueba: Verificar que se pueda actualizar los datos de una persona				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
UpdatePersonViewController	UpdatePersonViewController	setupStage4	Person person	Se actualiza los datos de la persona

Objetivo de la Prueba: Validar la correcta creación del archivo json				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
PersonData	saveJSON	setupStage3	*dos instancias del objeto persona.*	Se han cargado correctamente los datos en el archivo json.

Objetivo de la Prueba: Validar la correcta implementación de la información en el archivo json				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
PersonData	loadJSON	setupStage3	*ninguno*	Se han cargado correctamente los datos en el archivo json.

Objetivo de la Prueba: Validar la correcta asignación de un número aleatorio a la variable randomN.				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Controller	generaterandomNum	setupStage5	randomN=”2,36”	Se ha guardado un número aleatorio.

	berForPercent			

Objetivo de la Prueba: Seleccionar un nombre del archivo csv babynames-clean.csv				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Controller	randomValueCSV	setupStage6	relativePath=".\\data\\babynames-clean.csv"	Se ha seleccionado un nombre del archivo csv babynames-clean.csv

Objetivo de la Prueba: Seleccionar un nombre del archivo csv Apellidos.csv"				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
Controller	randomValueCSV	setupStage7	relativePath=".\\data\\Apellidos.csv"	Se ha seleccionado un apellido del archivo csv Apellidos.csv

DISEÑO DE PRUEBAS ÁRBOL ROJINEGRO

Objetivo de la Prueba: Agregar un nuevo nodo				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RedBlackTree	insert	setupStage9	element=Person();	Se ha agregado un nuevo nodo al árbol.

Objetivo de la Prueba: Buscar un nodo				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RedBlackTree	searchByCode	setupStage9	code="XMLU12"	Se ha encontrado el nodo.
RedBlackTree	searchByFullName	setupStage9	fullName="Cristian Perafan"	Se ha encontrado el nodo.
RedBlackTree	searchByName	setupStage9	name="Cristian"	Se ha encontrado el nodo.
RedBlackTree	searchByLastName	setupStage9	lastName="Perafan"	Se ha encontrado el nodo.

Objetivo de la Prueba: Eliminar un nodo				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RedBlack Tree	delete	setupStage9	node= node; node =current;	se ha eliminado al nodo correspondiente a la persona pasada por parametro

Objetivo de la Prueba: Balancear el árbol				
Clase	**Método**	**Escenario**	**Valores de Entrada**	**Resultado**
RedBlack Tree	leftRotate	setupStage9	RedBlackNode node=node;	Se ha balanceado hacia la izquierda el árbol
RedBlack Tree	rightRotate	setupStage9	RedBlackNode node=node;	Se ha balanceado hacia la derecha el árbol
RedBlack Tree	balanceAfterInsert	setupStage9	RedBlackNode node=node;	Se ha balanceado el arbol antes de insertar.