|  |  |  |
| --- | --- | --- |
| **Objetivos**   1. Realizar un ejercicio que permita incorporar java web 2. Almacenar información en la memoria temporal 3. Serializar la información de un aplicativo 4. Demostrar la comunicación en la web | **Número de participantes**  participante.jpgparticipante.jpgparticipante.jpg | **Materiales**   * Internet * Eclipse o Netbeans * Apache Tomcat * Maven |
| **Duración de la actividad**  **4:00** |
| **Proyecto:**  Desarrollo de una aplicación web para un Directorio de Contactos  **Autoría**   |  |  | | --- | --- | | Proyecto Curso: | Estructuras de datos I | | Ejercicio: | Aplicación web para directorio de contactos | | Autores: | 1.Karen Julieth Pérez | | 2.Omar Santiago Salazar | | Fecha realización: | 28-02-2024 |   **Listado de Requerimientos:**   |  |  | | --- | --- | | Nombre | R1- Agregar un contacto | | Resumen | Permitir al usuario agregar un nuevo contacto al directorio | | Entradas | nombre, apellido, correo electrónico, dirección y celular | |  |  | | Resultados | Se muestra la lista de los perros inscritos en el desfile | |  | |  |  |  | | --- | --- | | Nombre | R2- Ver los datos de un contacto | | Resumen | Mostrar la información de un contacto específico | | Entradas | N/A | |  |  | | Resultados | Se muestra el nombre, apellido, correo, dirección y celular del contacto exitosamente | |  | |  |  |  | | --- | --- | | Nombre | R3- Buscar un contacto | | Resumen | Permitir al usuario buscar un contacto por su nombre | | Entradas | Nombre | |  |  | | Resultados | Se encuentra el contacto en el árbol binario y se muestran sus datos | |  | |  |  |  | | --- | --- | | Nombre | R4- Eliminar un contacto | | Resumen | Permitir al usuario eliminar un contacto del directorio | | Entradas | N/A | |  |  | | Resultados | El contacto se encuentra y se elimina exitosamente | |  | |   **Modelo conceptual**    ***Requisitos Técnicos:***   1. Este proyecto web permitirá a los usuarios agregar, eliminar y ver un directorio de contactos utilizando una estructura de árbol binario. El directorio se organizará en base a los nombres agregados, y cada contacto tendrá datos como (DNI, Nombre, Apellido, Celular, Correo electrónico). A continuación, se definirán datos sobre cómo se configurará, ejecutará este proyecto web.   **Entregables Esperados:**  1. Debe crear un repositorio en GIT para el proyecto  2. Documentación que explique la estructura del proyecto, boceto de la interfaz de usuario  3. Capturas de pantalla que muestren la aplicación en funcionamiento  **Criterios de Evaluación:**  Los proyectos serán evaluados en función de la correcta implementación de los requisitos funcionales y técnicos, la aplicación adecuada de la programación orientada a objetos, así como la usabilidad y apariencia de la interfaz web. Los estudiantes también deben ser capaces de explicar y justificar las decisiones de diseño y tecnología que tomaron durante el desarrollo del proyecto. Con ello:   1. Todos los requisitos funcionales desarrollados 2. Utilización de la programación orientada a objetos 3. Funcionalidad, el programa debe funcionar en su totalidad sin ningún error 4. Usabilidad, el programa debe ser fácil de utilizar, no debe dejar dudas al usuario 5. El archivo en un repositorio GIT de forma adecuada 6. Código documentado en su totalidad 7. Puntualidad en la entrega 8. Esta documentación completa con el prototipo de la interfaz   **Prototipo de interfaz**    Documentación del proyecto   * Vamos a definir una clase infContacto la cual va a representar la información de un contacto, con la información necesaria la cual es id, nombre, apellido, correo, dirección y numero de Celular. * Dentro de la clase infContacto también van a estar los nodos que representaran a un árbol binario. Cada nodo tendrá un Contacto y referencias a sus nodos hijos izquierdo y derecho. * Implementaremos una clase Arbol\_b que va a contener la lógica para insertar contactos en el árbol, validar la raíz y delegar la solución al contacto con la raíz y cumplir su rol, organizando los contactos por nombre para hacer más fácil su visualización. * Definimos una clase métodos para hacer que las funciones requeridas sean funcionales al momento de usarlos  1. ***Clase infContacto***   La clase infContacto implementa la Comparable interfaz, que permite ordenar las instancias de esta clase. La clase tiene las siguientes propiedades:  id, nombre, apellido, correo, dirección y teléfono: Cadenas que almacenan la información del contacto.  Izquierdo y derecho: Referencias a los nodos secundarios izquierdo y derecho, respectivamente, que forman el árbol de búsqueda binario.  La clase proporciona varios métodos para manipular y consultar el árbol AVL:  **Constructores:** Dos constructores, uno sin parámetros y otro con toda la información de contacto como parámetros.  compareTo (): Implementa el método Comparable de la interfaz, comparando la propiedad de dos instancias.compareTo()nombreinfContacto.      **Buscar(String Nombres):** busca un contacto por su nombre en todo el árbol AVL y devuelve el contacto si lo encuentra.  **borrar(String Nombre):** Elimina un contacto por su nombre del árbol AVL y mantiene el equilibrio.  **Hoja(), Menor(), Mayor(), Altura(), peso\_I(), Peso()y NumHojas():** métodos de utilidad para obtener información sobre el árbol AVL, como altura, peso y número de nodos de hojas.  La clase utiliza algoritmos de árbol AVL para mantener el equilibrio durante las operaciones de inserción y eliminación. El equilibrio se mantiene mediante rotaciones (izquierda y derecha) y rotaciones dobles (izquierda-derecha y derecha-izquierda) para garantizar que la diferencia de altura entre los subárboles izquierdo y derecho sea como máximo uno.  Esta clase infContacto puede resultar útil para almacenar información de contacto de forma estructurada y organizada y, al mismo tiempo, proporcionar operaciones rápidas de búsqueda, inserción y eliminación   1. ***Clase Arbol\_b***   La clase Arbol\_b  tiene un contacto raíz (contacRaiz) y un contador de contactos ( listaContac). Los contactos se almacenan en el árbol según sus nombres, lo que permite operaciones rápidas de búsqueda, inserción y eliminación.  La clase tiene los siguientes métodos:  **Constructor:** Inicializa el contacto raíz en nulo y establece el contador de contactos en 0.  **listaDeContactos :** Devuelve un ArrayList que contiene todos los contactos del árbol. Lo hace llamando recursivamente a un método privado guardarNuevoContactoL que almacena los contactos en ArrayList.  **guardarNuevoContactoL**: Un método recursivo privado que almacena los contactos en un ArrayList. Es llamado por listaDeContactos.  **contactoYaGuardado**: Un método privado que verifica si un contacto con un nombre determinado ya existe en el árbol. Lo hace buscando recursivamente en el árbol.  **guardarContacto**: Inserta un nuevo contacto en el árbol. Primero verifica si el contacto ya existe en el árbol usando el contactoYaGuardadométodo. Si el contacto ya existe, arroja un archivo ContactoExistenteException. Si el árbol está vacío, el nuevo contacto se convierte en la raíz. De lo contrario, llama al ingresarmétodo del contacto raíz para insertar el nuevo contacto en el árbol. Luego incrementa el contador de contactos.  **borrarContac**: Elimina un contacto del árbol. Lo hace llamando al borrarmétodo del contacto raíz. Luego disminuye el contador de contactos.  **buscarC**: Busca un contacto en el árbol según su nombre. Lo hace llamando al Buscarmétodo del contacto raíz.  **Peso**: Devuelve el número total de contactos en el árbol llamando al Pesométodo del contacto raíz. Si el árbol está vacío, devuelve 0.  Esta clase utiliza composición para almacenar los contactos en el árbol y ofrece un buen equilibrio entre encapsulación y facilidad de uso. Sigue buenas prácticas para el manejo de excepciones y mantiene una estructura limpia y organizada.     1. ***Clase métodos***   ***La clase tiene dos métodos:***  **añadirContacto(Arbol\_b contac, ServletContext context) throws IOException**: Este método se utiliza para agregar un contacto a un archivo llamado contac.ser. Aquí Arbol\_bse supone que una clase personalizada maneja datos de contacto.  Primero genera la ruta absoluta del archivo a partir de la ruta relativa y crea un nuevo Fileobjeto para el archivo de datos.  Usando ObjectOutputStream, escribe el Arbol\_bobjeto en el archivo.  Si IOExceptionocurre algo durante este proceso, captura e imprime el seguimiento de la pila junto con un mensaje de error.  **cargarContactos(ServletContext context) throws IOException, ClassNotFoundException**: Este método se utiliza para cargar los contactos del contac.serarchivo en un Arbol\_bobjeto.  Genera la ruta absoluta del archivo a partir de la ruta relativa y crea un nuevo Fileobjeto para el archivo de datos.  Usando ObjectInputStream, lee el Arbol\_bobjeto del archivo.  Si ocurre algo IOExceptiono ClassNotFoundExceptiondurante este proceso, captura e imprime el seguimiento de la pila junto con un mensaje de error si el Arbol\_bobjeto cargado es null, inicializa un nuevo Arbol\_bobjeto.  Finalmente, devuelve el Arbol\_bobjeto cargado o inicializado.  Es probable que estos métodos se utilicen en el contexto de una aplicación web, dado el uso de ServletContext. Los métodos pueden resultar útiles para desarrolladores Java principiantes y avanzados que trabajan con manejo de archivos, serialización de objetos y desarrollo de aplicaciones web.     1. ***Servlet SvDirectorio***   El servlet recibe un nombre SvDirectorio y se asigna al patrón de URL /SvDirectorio. Extiende la HttpServletclase y anula los métodos doGet y doPost para manejar diferentes tipos de solicitudes HTTP.  El método doGet se utiliza para manejar solicitudes GET. Primero obtiene el contexto del servlet y recupera un atributo denominado arbol\_b, que representa un árbol de contactos. Si el atributo es null, redirige al usuario a la página de índice. Luego, verifica la presencia de un parámetro de solicitud denominado action. Si el parámetro está configurado en "eliminarC", recupera el nombre del contacto que se va a eliminar, lo imprime en la consola e intenta eliminarlo del árbol de contactos. Si la eliminación se realiza correctamente, actualiza la lista de contactos en el contexto y los atributos de la sesión, y redirige al usuario a la página de índice. Si hay un error durante la eliminación, establece un mensaje de error en el atributo de solicitud y reenvía al usuario a la página de índice.    El método doPost se utiliza para manejar solicitudes POST. Primero obtiene el contexto del servlet y recupera el árbol de contactos del atributo de contexto. Si el atributo es null, crea un nuevo árbol de contactos y lo establece en el atributo de contexto. Luego, recupera los datos del formulario enviados por el usuario, crea un nuevo objeto de contacto con los datos e intenta guardarlo en el árbol de contactos. Si el contacto ya existe, establece un mensaje de error en el atributo de sesión y redirige al usuario a la página de índice. Si el contacto se guarda correctamente, actualiza la lista de contactos en el contexto y los atributos de la sesión, establece un mensaje de éxito en el atributo de la sesión y redirige al usuario a la página de índice.  El servlet también define un método privado llamado lista que toma un árbol de contactos como argumento y devuelve una lista de objetos de contacto del árbol. Este método se utiliza en el doGetmétodo para recuperar la lista de contactos actualizada después de una eliminación. | | |