UNIVERSIDAD DE LAS FUERZAS ARMADAS "ESPE"

INTEGRANTES:

- ***Bonifaz Christian***
- ***Calderon Mateo***
- ***Guano Ariel***
- ***Mera Erick***
- ***Rosas Mario***

FECHA: 05/07/2021

CARRERA:

Electrónica y Automatización

NRC:

3725

ASIGNATURA:

Programación Orientada a Objetos

TEMA:

Persistencia de Datos en JAVA

OBJETIVOS:

1. Seguir avanzando dentro del campo de la programación orientada a los objetos con el estudio y aplicación de persistencia de datos en JAVA y la creación y lectura de archivos planos en java
2. Analizar y comprender el uso de la persistencia de datos para aplicarlos en nuestro en el cual estos datos sobreviven de alguna manera siendo almacenados y que no sean eliminados.
3. Analizar y comprender el uso de la lectura de archivos planos para aplicarlos en nuestro programa para crear o abrir un archivo.

MARCO TEÓRICO:

Persistencia de datos en Java

La idea de trabajar con entidades persistentes ha estado presente en la Programación Orientada a Objetos desde sus comienzos. Este enfoque intenta aplicar las ideas de la POO a las bases de datos, de forma que las clases y los objetos de una aplicación puedan ser almacenados, modificados y buscados de forma eficiente en unidades de persistencia. Sin embargo, aunque desde comienzos de los 80 hubo aplicaciones que implementan bases de datos orientadas a objetos de forma nativa, la idea nunca ha terminado de cuajar. La tecnología dominante en lo referente a bases de datos siempre han sido los sistemas de gestión de bases de datos relacionales (RDBMS). De ahí que la solución propuesta por muchas tecnologías para conseguir entidades persistentes haya sido realizar un mapeado del modelo de objetos al modelo relacional. JPA es una de estas tecnologías.

La persistencia de datos es un medio mediante el cual una aplicación puede recuperar información desde un sistema de almacenamiento no volátil y hacer que esta persista. La persistencia de datos es vital en las aplicaciones empresariales debido al acceso necesario a las bases de datos relacionales. Las aplicaciones desarrolladas para este entorno deben gestionar por su cuenta la persistencia o utilizar soluciones de terceros para manejar las actualizaciones y recuperaciones de las bases de datos con persistencia. JPA (Java™ Persistence API) proporciona un mecanismo para gestionar la persistencia y la correlación relacional de objetos y funciona desde las especificaciones EJB 3.0.

Los conceptos básicos de JPA son:

• Entidades persistentes

• Mapeo entidad-relación

• Relaciones entre entidades

• Entity manager y transacciones

JPA está diseñado para funcionar dentro y fuera de un contenedor Java Enterprise Edition (Java EE). Cuando se ejecuta JPA dentro de un contenedor, las aplicaciones pueden utilizar el contenedor para gestionar el contexto de persistencia. Si no hay ningún contenedor para gestionar JPA, la aplicación debe manejar ella misma la gestión del contexto de persistencia. Las aplicaciones diseñadas para la persistencia gestionada por contenedor no requieren tanta implementación de código para manejar la persistencia, pero estas aplicaciones no se pueden utilizar fuera de un contenedor. Las aplicaciones que gestionan su propia persistencia pueden funcionar en un entorno de contenedor o en un entorno Java SE.

Creación y lectura de archivos planos en java

- **Creación de un archivo**

Para crear un archivo en Java podemos usar dos técnicas diferentes: La primera es usando las clases FileWriter y BufferWriter, donde usamos el método write que te permite escribir cadenas o arreglos de caracteres y la segunda forma es usando PrintWriter que te permite hacer más o menos lo mismo, pero de una forma más resumida y con la posibilidad de escribir otros tipos de datos sobre el archivo.

Cabe recalcar que la clase FileWriter debe crearse con una referencia a una clase File que contiene los detalles del archivo que será creado y el contenido del texto se crea con la función bw.write(contenido) de BufferedWriter y dependerá de cada persona el agregar el carácter de salto de línea.

Otra opción para crear archivos es mediante el uso de la clase PrintWriter que puede recibir una cadena con la ruta del archivo o una instancia de File y un segundo argumento que es el tipo de enconding, que será útil si necesitamos guardar texto con caracteres especiales como acentos y eñes.

- **Lectura de un archivo**

En Java existen diversas formas de leer un archivo, pero una de las más sencillas es usando la clase Scanner.

También podemos leer un archivo usando la clase FileReader. Esta clase tiene métodos que nos permiten leer caracteres. Sin embargo, suele ser habitual querer las líneas completas, bien porque nos interesa la línea completa, bien para poder analizarla luego y extraer campos de ella. FileReader no contiene métodos que nos permitan leer líneas completas, pero sí BufferedReader. Afortunadamente, podemos construir un BufferedReader a partir del FileReader.

La apertura de un archivo y su posterior lectura pueden lanzar excepciones que debemos capturar. Por ello, la apertura del archivo y la lectura debe meterse en un bloque try-catch.

Además, el archivo hay que cerrarlo cuando terminemos con él, tanto si todo ha ido bien como si ha habido algún error en la lectura después de haberlo abierto. Por ello, se suele poner al try-catch un bloque finally y dentro de él, el close() del archivo.

CONCLUSIONES Y RECOMENDACIONES:

1. Se concluye que mediante la creación de archivos planos y lectura de archivos Java más la complementación del Apache Netbeans es de gran utilidad ya que nos permite desde el punto de vista tener un archivo de respaldo en donde se va registrando en una hoja de texto todos nuestros resultados obtenidos en el código que hayamos programado.

2. Una de las conclusiones también que llegamos fue, que al salir de un programa los datos generados quedarán guardados en algún lugar que permitiera su posterior recuperación desde el mismo u otros programas. Por tanto, dichos datos quedarán guardados.

3. Se concluye que la implementación de archivos de texto en los programas que estamos desarrollando da un perfil un poco más profesional y más estético para el desarrollo del programa y del trabajo que estamos realizando.

BIBLIOGRAFÍA:

Delgado, J. (2021). Crear archivos de texto. Obtenido de: https://decodigo.com/java-crear-archivos-de-texto

Groussard, T. (2012). Los fundamentos del lenguaje Java. Obtenido de: https://books.google.es/books?hl=es&lr=&id=JaPTzKZxbN4C&oi=fnd&pg=PA9&dq=creacion+de+archivos+planos+en+java&ots=pW1GrhukXh&sig=xBRyqG97DzEN6pM03pm2E4F3QtA#v=onepage&q&f=false

Chuwuki. (2017). Lectura y escritura de archivos en Java. Obtenido de: http://chuwiki.chuidiang.org/index.php?title=Lectura_y_Escritura_de_Ficheros_en_Java

IBM Docs. Ibm.com. (2021). Retrieved 5 July 2021, Obtenido de: https://www.ibm.com/docs/es/was-liberty/base?topic=overview-java-persistence-api-jpa.

Jtech.ua.es. (2021). Retrieved 5 July 2021, from http://www.jtech.ua.es/j2ee/2010-2011/restringido/jpa/wholesite.pdf.