UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
CENTRO UNIVERSITARIO DE OCCIDENTE
DIVISIÓN DE CIENCIAS DE LA INGENIERÍA
MANEJO E IMPLEMENTACIÓN DE ARCHIVOS

MANUAL TÉCNICO

DIEGO ALESSANDRO GONZALEZ BRAVO / 202230939

Tecnologías utilizadas:

- Java 21: Como lenguaje de programación orientado a objetos
- Inteliji: Como ide utilizado.

Estructura de proyecto:

Models: Maneja los objetos

- Libro: El objeto libro en el cual lo podemos crear desde 0 o importando un archivo
- Préstamo: El objeto que nos registra un préstamo de un libro

Reports: Maneja los reportes

- Reportes: Manejo de una lista de libros que nos ayuda a poder mostrar los reportes que se generan utilizando la aplicación

Services: Manejo de los servicios los cuales nos sirven para el uso de la aplicacion

- Biblioteca: Manejo de la lista de libros ya sea que se ingresa o que se obtiene de los archivos de persistencias
- LibroService: Con este servicio obtiene la lista de libros serializados por un archivo de persistencia o crea una lista de libros con un archivo de entrada, también actualiza el manejo de los archivos de persistencia.
- PathArchivos: Con este servicio obtenemos el path relativo de en donde se encuentra el .jar de la aplicación
- PrestarService: Con este manejamos los archivos de persistencia sobre los préstamos que se realizaron

Implementación de serializable

Para el manejo de objetos serializables se implementó la función de java que nos permite seriabilizarlos:

- public class Libro implements Serializable
- public class Prestamo implements Serializable

Implementación de getters y setters

Con el uso de las dependencias el pom, se utilizo la dependencia de lombok con la cual nos facilita implementaciones de codigo para los getters y setters de un objeto, codigo de la dependencia:

```
<dependencies>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.38</version>
    </dependency>
  </dependencies>
```

Escritura en archivos binarios:
Al estar serializados los objetos a la hora de guardarlos en los archivos binarios se tienen que instanciar al objeto que los necesitemos para su mejor manejo, este es un pedaso de codigo con el cual se puede guiar para el manejo de este caso es el de prestamos.gbr:

```
private void saveInFile() {
          try    (ObjectOutputStream    objectOutputStream    =    new
ObjectOutputStream(new FileOutputStream(filePath))) {
       objectOutputStream.writeObject(this.prestamos);
   } catch (IOException e) {
             System.out.println("NO  SE  PUDO  GUARDAR  LA  LSITA"  +
e.getMessage());
   }
}

private List<Prestamo> returnprestamos(Object object) {
   if (!(object instanceof List<?>)) {
       return null;
   }
   final List<?> list = (List<?>) object;
   for (Object item : list) {
       if (!(item instanceof Prestamo)) {
           return null;
       }
   }
   return (List<Prestamo>) list;
}

private void loadData() {
   final File file = new File(filePath);
   if (file.exists()) {
                try   (ObjectInputStream   objectInputStream   =   new
ObjectInputStream(new FileInputStream(filePath))) {
           final Object object = objectInputStream.readObject();
           final List<Prestamo> list = returnprestamos(object);
           if (list == null) {
               System.out.println("NO ES UNA LISTA DE PRESTAMOS");
               return;
           }
           this.prestamos = list;
                    System.out.println("SE  CARGARON  LOS  PRESTAMOS
CORRECTAMENTE");
       } catch (Exception e) {
            System.out.println("OCURRIO UN ERROR A LA HORA DE CARGAR
DATOS: " + e.getMessage());
       }
   } else {
       System.out.println("EL ARCHIVO DE PRESTAMOS NO EXISTE");
   }

}
```

Obtener path relativo:
Para obtener el path relativo del programa se utilizo una función de java tipo File, este es el codigo que se utilizo para el funcionamiento adecuado y se utilizo la dependencia de lombok:

```
@Getter
private String pathArchivo;

public pathArchivos() {
    this.pathArchivo = getExecutionPath();

}

public String getExecutionPath() {
    try {
        String path = new java.io.File(
                Biblioteca.class.getProtectionDomain()
                        .getCodeSource()
                        .getLocation()
                        .toURI()
        ).getParent();

        return path;
    } catch (Exception e) {
        System.out.println("NO SE OBTUVO EL PATH DEL PROGRAMA");
        return ".";
    }
}
```