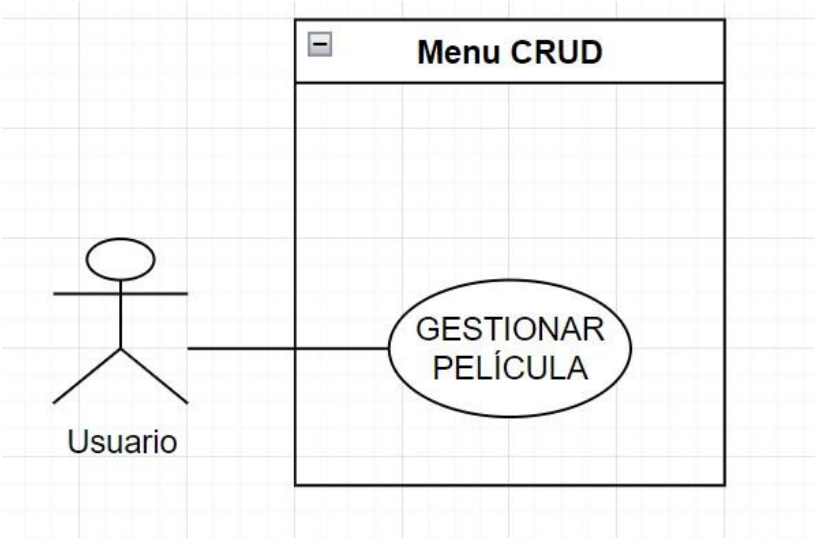
Desarrollo de programa CRUD con Java

Alfonso López Delgado

1ºDAM

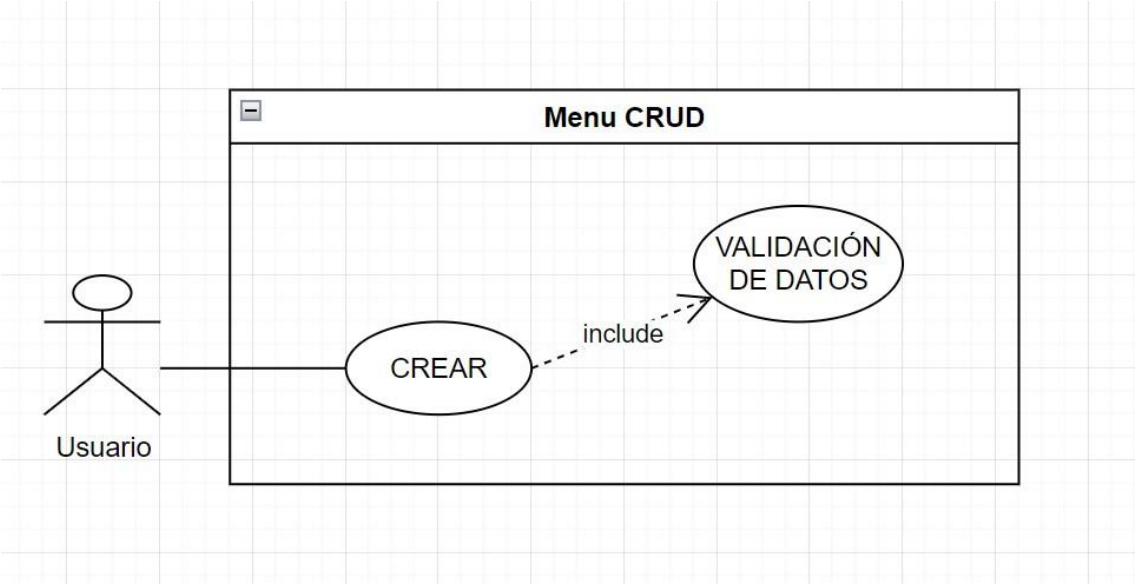
Análisis de casos de uso CRUD tema películas

1. Gestionar película



NOMBRE		DESCRIPCIÓN
Gestionar película		El usuario puede elegir entre 5 opciones para tratar los datos, crear, modificar, borrar, consultar y salir del menú
PRECONDICIONES		POSTCONDICIONES
• Iniciar programa • Mostrar menú • Introducción por teclado		• Dirección opciones
DATOS	DATOS ENTRADA	DATOS SALIDA
• Mostrar menú	• Mostrar menú • Dato	• Mostrar opción

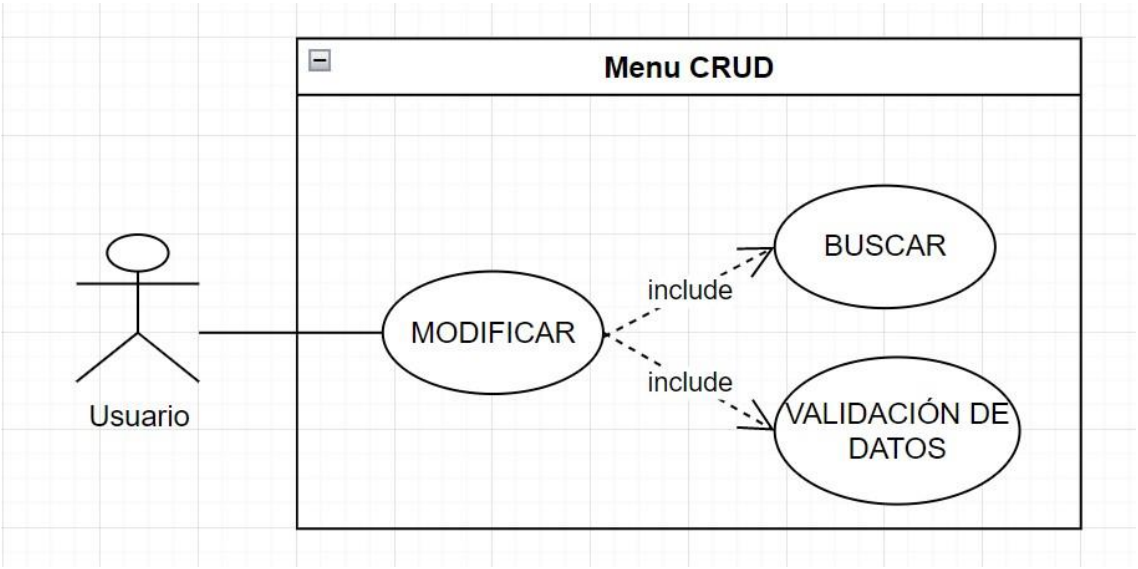
2. Crear



NOMBRE		DESCRIPCIÓN
Crear película		El usuario crea una película con los datos que se le pide
PRECONDICIONES		POSTCONDICIONES
• Array para almacenar el nombre • Array para almacenar la fecha • Array para almacenar la valoración • Introducción por teclado de película		• Película nueva previa validación de datos.
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Array fecha • Array valoración	• Mensaje	• Dato (nombre, fecha, valoración)

NOMBRE		DESCRIPCIÓN
Validación de datos		Validación de datos introducidos por el usuario
PRECONDICIONES		POSTCONDICIONES
• Dato introducido por teclado • Dato dentro de los parámetros		• Dato insertado en los arrays dentro de los parámetros establecidos
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Dato	• Mensaje • Dato sin validar	• Dato (nombre, fecha, valoración) validado

3. Modificar

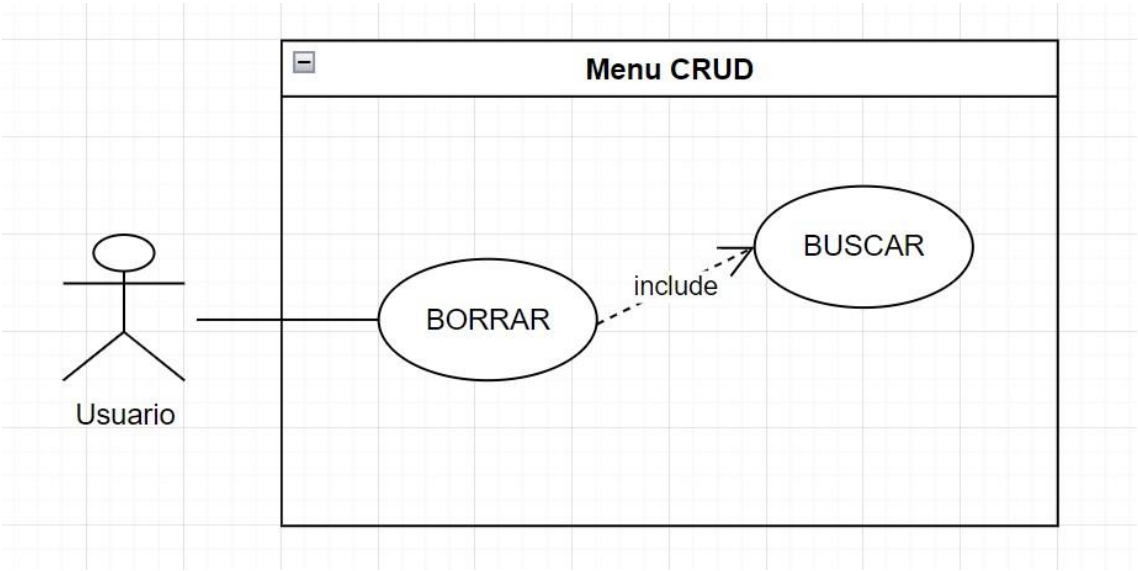


NOMBRE		DESCRIPCIÓN
Modificar		Modificación de algún dato, que haya introducido el usuario previamente.
PRECONDICIONES		POSTCONDICIONES
• Dato almacenado en Array • Introducción del nuevo dato		• Eliminación del antiguo dato y sustitución por el nuevo
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Dato	• Mensaje	• Dato (nombre, fecha, valoración) nuevos

NOMBRE		DESCRIPCIÓN
Buscar		Buscar el dato introducido por el usuario
PRECONDICIONES		POSTCONDICIONES
• Dato introducido por teclado • Dato almacenado en Array		• Dato encontrado
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Dato posición	• Dato teclado	• Dato posición

NOMBRE		DESCRIPCIÓN
Validación de datos		Validación de datos introducidos por el usuario
PRECONDICIONES		POSTCONDICIONES
• Dato introducido por teclado • Dato dentro de los parámetros		• Dato insertado en los arrays dentro de los parámetros establecidos
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Array fecha • Array valoración • Dato	• Mensaje • Dato sin validar	• Dato (nombre, fecha, valoración) validado

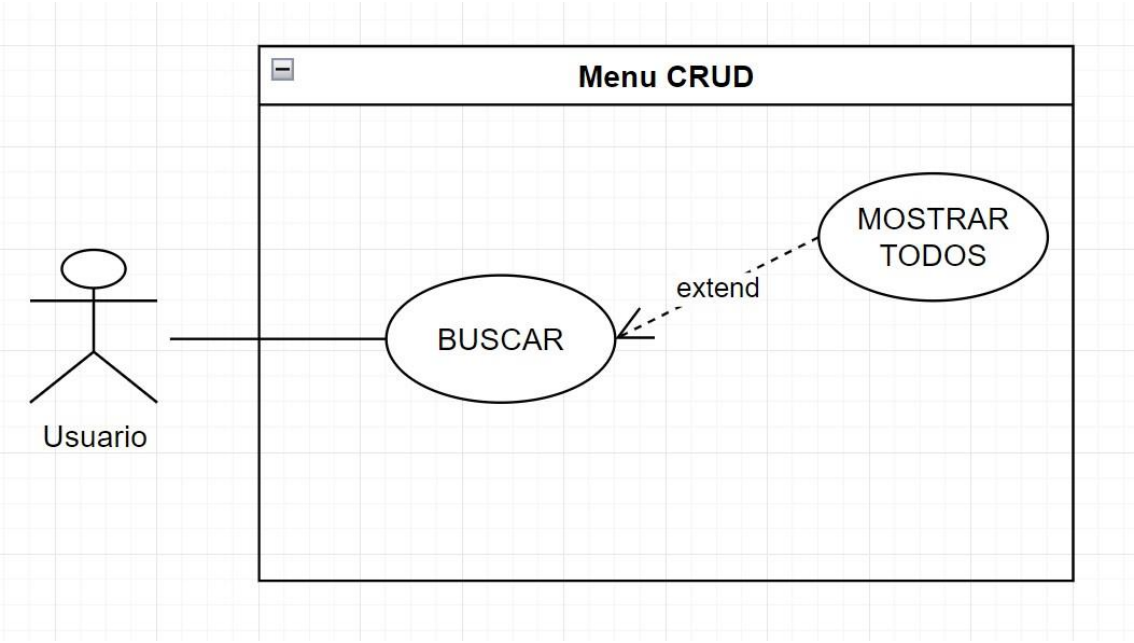
4. Borrar



NOMBRE		DESCRIPCIÓN
Borrar		Borrado de un dato en el array mediante la sustitución de otro dato neutral.
PRECONDICIONES		POSTCONDICIONES
• Dato almacenado en Array		• Sustitución por elemento neutral (como un espacio)
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre	• Mensaje	• Dato (“”, “”, “”)

NOMBRE		DESCRIPCIÓN
Buscar		Buscar el dato introducido por el usuario
PRECONDICIONES		POSTCONDICIONES
• Dato introducido por teclado • Dato almacenado en Array		• Dato encontrado
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Dato posición	• Dato teclado	• Dato posición

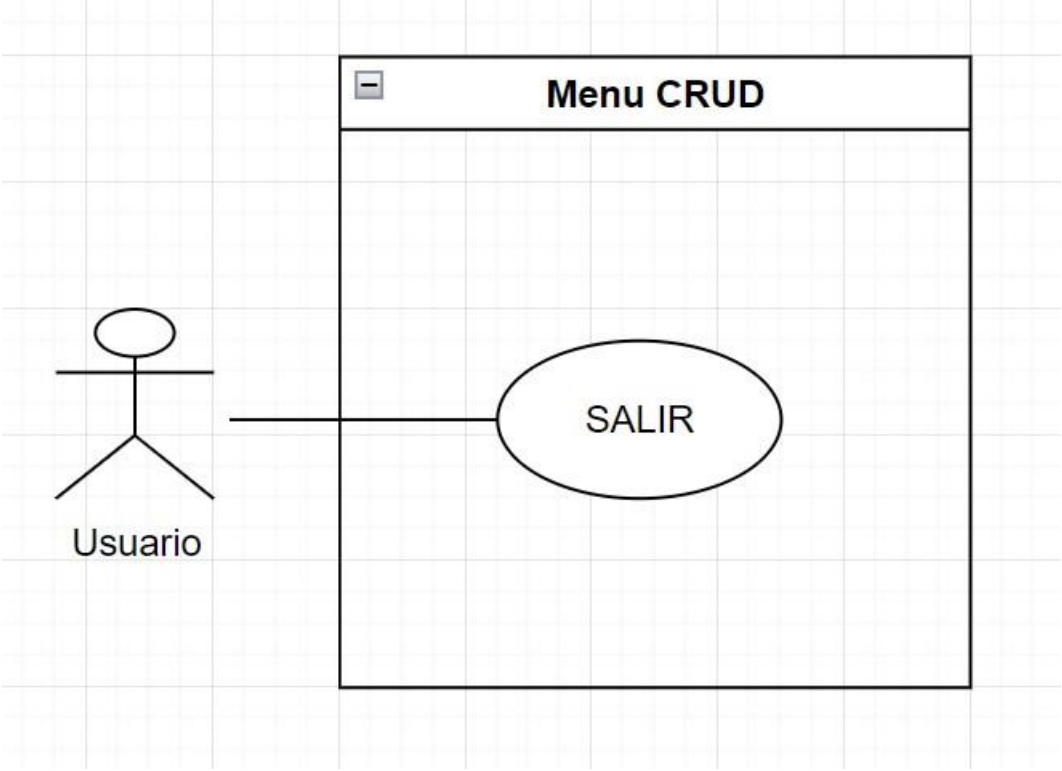
5. Buscar



NOMBRE		DESCRIPCIÓN
Buscar		Buscar el dato introducido por el usuario
PRECONDICIONES		POSTCONDICIONES
• Dato introducido por teclado • Dato almacenado en Array		• Dato encontrado • Mensaje
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre • Dato posición	• Dato teclado	• Mensaje

NOMBRE		DESCRIPCIÓN
Mostrar todos		El usuario puede elegir si hacer una búsqueda específica o mostrar todos los datos del array
PRECONDICIONES		POSTCONDICIONES
• Elegir opción mostrar todo		• Mostrar todos los datos del array
DATOS	DATOS ENTRADA	DATOS SALIDA
• Array nombre		• Mensaje

6. Salir



NOMBRE		DESCRIPCIÓN
Salir		Opción salir del menú
PRECONDICIONES		POSTCONDICIONES
• Elegir opción		• Mensaje • Apagar consola
DATOS	DATOS ENTRADA	DATOS SALIDA
		• Mensaje

RFTP

T01	Elección de opción Menú						
		F01	Mostrar Opciones				
				T01	Crear diseño de menú		
						P01	Probar con la consola que se ve correcto
				T02	Elegir una opción		
						P01	Probar que funciona la opción de elegir opción
T02	Menu CRUD						
		F01	Opción crear película				
				T01	Creación de tres Arrays (Nombre, año y valoración)		
						P01	Probar que los datos se guardan correctamente
				T02	Introducción de datos por teclado		
						P01	Probar introducción de datos con la consola
						P02	Probar que se guardan en las posiciones adecuadas
				T03	Validación de datos		
						P01	Probar que los datos introducidos están dentro de los parámetros establecidos
				T04	Opción de volver a introducir una película antes de salir al Menu		
						P01	Probar que funciona el bucle cuando introduces la letra 's' o 'n'
						P02	Probar que si introduces una letra diferente sale mensaje de error
		F02	Opción modificar película				
				T01	Buscar película mediante herramienta consultar película		
						P01	Probar que se introducen datos por teclado
						P02	Probar que los datos introducidos coinciden con los ya existentes en los Arrays
						P03	Probar que si da error sale el mensaje
				T02	Introducción de datos nuevos por teclado		
						P01	Probar introducción de datos con la consola
						P02	Probar que se guardan en las posiciones adecuadas
				T03	Validación de datos nuevos		
						P01	Probar que los datos nuevos introducidos están dentro de los parámetros establecidos
		F03	Opción borrar película				
				T01	Buscar película mediante herramienta consultar película		
						P01	Probar que se introducen datos por teclado
						P02	Probar que los datos introducidos coinciden con los ya existentes en los Arrays
				T02	Sustituir dato por el elegido para que parezca un borrado		
						P01	Probar que los datos se sustituyen correctamente
		F04	Opción de consultar				
				T01	Consulta genérica		
						P01	Probar que se introducen datos por teclado
						P02	Recorrer todo el Array y comprobar que se muestra por consola correctamente
				T02	Consulta específica		
						P01	Probar que se introducen datos por teclado
						P02	Recorrer el Array hasta encontrar el dato y comprobar que se muestra por consola corre
		F05	Mostrar consulta				
				T01	Mostrar consulta previa		
						P01	Probar que se muestra correctamente la consulta
		F06	Salir del programa				
				T01	Cerrar el programa		
						P01	Probar que cierra el programa cuando se elige la opción

RFTP

Código comentado

Este apartado será una breve explicación del código del programa que se ha realizado.

Clase utilidades

Es creado con la intención de la reutilización del código que se repetirá en reiteradas ocasiones durante el proyecto.

- **Método pedirString()** → Creado con la finalidad de introducir por teclado los valores String que necesite el usuario.

  Explicación del código: Creación de método donde retornara un dato String pedida en la variable "dato" mediante la clase Scanner.

```java
/**
 * Introducción de datos por teclado
 *
 * @return String
 */
public static String pedirString() {

    String dato;
    Scanner lector = new Scanner(System.in);

    dato = lector.nextLine();

    return dato;

}
```

- **Método confirmación()** → Creado con la finalidad de seguir en la misma opción del menú si el usuario lo requiere y evitar salir y volver a entrar.

  Explicación del código: Creación de un método donde retornara un dato char. Se pedirá una letra con la variable tipo char "letra" con la clase Scanner. El dato introducido entrará en un bucle while donde se repetirá mientras la letra sea distinto de 's' o 'n', unidos por un operador lógico AND, y lanzará un mensaje de "error".

```java
/**
 * Sentencia repetitiva para evitar volver al menú principal donde solo deja
 * introducción por teclado de dos valores
 *
 * @return char s o n
 * @param texto Lanza un texto por pantalla
 */

public static char confirmacion(String texto) {
    char letra = 0;
    Scanner lector = new Scanner(System.in);

    System.out.println(texto);
    letra = lector.next().charAt(0);

    while (letra != 's' && letra != 'n') {
        System.out.println("No se reconoce, vuelva a introducir (n o s)");
        letra = lector.next().charAt(0);
    }
    return letra;
}
```

- **Método pedirNumMenu()** → Creado con la finalidad de pedir una opción al usuario para elegir en el menú.

  Explicación de código: Creación de método en el cual retornara un dato int. Se pedirá un número recogido con la variable "num", de tipo int, por teclado mediante la clase Scanner. El dato entra en un bucle while el cual se repetirá mientras dichos parámetros se cumplan, unidos por un operador lógico OR, y lanzará un mensaje por pantalla de "error".

```
54    /**
55     * Introducción mediante teclado de un número para el menú. Comprobación de
56     * dicho número este dentro de los parametros indicados.
57     *
58     * @return int
59     *
60     */
61    public static int pedirNumMenu() {
62
63        int num;
64        Scanner lector = new Scanner(System.in);
65
66        num = lector.nextInt();
67        lector.nextLine();
68
69        while (num < 1 || num > 5) {
70            System.out.println("Error vuelva a elegir una opcion");
71            num = lector.nextInt();
72            lector.nextLine();
73        }
74        return num;
75
76    }
77 }
```

Clase menuFinal

- **Método menú()** → La estética del menú que se imprimirá por consola.

  Explicación del código: Mediante el conjunto de instrucciones System.out.println(""); se le dará una forma gráfica.

```
80    /**
81     * Estética menú
82     */
83
84    private static void menu() {
85        System.out.println("-------------------------");
86        System.out.println("| 1 |    Anadir     |");
87        System.out.println("-------------------------");
88        System.out.println("| 2 |   Modificar   |");
89        System.out.println("-------------------------");
90        System.out.println("| 3 |    Borrar     |");
91        System.out.println("-------------------------");
92        System.out.println("| 4 |   Consultar   |");
93        System.out.println("-------------------------");
94        System.out.println("| 5 |     Salir     |");
95        System.out.println("-------------------------");
96
97    }
98
```

- **Método inicializar(String[] lista, String[] lisFe, String[] lisVal, String dato)** → Su finalidad es rellenar todas las posiciones de los tres arrays lista, lisFe, lisVal con un dato. Referencia: Esta parte del código está sacada de las clases.

  Explicación del código: Creación de un método donde no devolverá nada y le dirá al programa que rellene los parámetros declarados. Mediante la variable "longitud" y la función "length" recogeremos la longitud del array y con un bucle for recorreremos todo el array introduciendo en cada posición del array el parámetro "dato".

```
256     /**
257      * Método que inicializa lista, lisFe, lisVal con lo introducido en el
258      * paramatetro dato
259      *
260      *
261      * @param lista  Array lista para alamacenar peliculas.
262      * @param lisFe  Array lista para alamacenar fecha.
263      * @param lisVal Array lista para alamacenar la valoración.
264      * @param dato   String de inicialización
265      *
266      */
267
268     public static void inicializar(String[] lista, String[] lisFe, String[] lis
269         int longitud = lista.length;
270
271         for (int i = 0; i < longitud; i++) {
272             lista[i] = dato;
273             lisFe[i] = dato;
274             lisVal[i] = dato;
275         }
276     }
277
278 }
```

- **Método buscarLibre(String[] lista, String[] lisFe, String[] lisVal, String dato)** → Su finalidad es buscar la primera posición del array donde haya un hueco libre. Referencia: Esta parte del código está sacada de las clases.

  Explicación del código: Creación de un método donde retornará una Int el cual será la posición donde se localiza el primer hueco libre. Mediante un bucle while con la condición de que se ejecute mientras la variable contador sea menor que la longitud del array AND la variable booleana distinta de la declarada, y anidando una sentencia if comparando cada posición del array con el "dato". Cuando la variable booleana es "true" sale del while con la variable "posición" igual al número de vueltas que ha dado el contador y retornará la posición donde se ha encontrado el primer "dato".

```
100      * Método para calcular la próxima posición del array sin datos guardados
101      *
102      *
103      * @param lista  Array lista para alamacenar peliculas
104      * @param lisFe  Array lista para alamacenar fecha
105      * @param lisVal Array lista para alamacenar la valoración
106      * @param dato   String de inicializacion
107      *
108      * @return int posición
109      */
110     public static int buscarLibre(String[] lista, String[] lisFe, String[] lisV
111         int posicion = -1;
112         int longitud = lista.length;
113         boolean encontrado = false;
114         int contador = 0;
115
116         /*
117          * Bucle para encontrar el siguiente valor, donde en el main lo inicial
118          * ""
119          */
120         while (contador < longitud && !encontrado) {
121             if (lista[contador].equals(dato)) {
122                 encontrado = true;
123                 posicion = contador;
124             }
125             contador++;
126         }
127
```

- **Método buscar(String[] lista, String[] lisFe, String[] lisVal) →** Su finalidad es buscar la posición del dato introducido por el usuario.

  Explicación del código: Mediante los parámetros lista, lisFe, lisVal.

  i. Solicitamos la clase utilidades.pedirString(); creada previamente, se pedirá un dato tipo String que se guardará en la variable "buscar".
  ii. Utilizaremos un bucle do while con la condición, mientras la variable "contador" sea menor a la longitud del array AND la variable "encontrado" distinta a la declarada.
  Anidando una sentencia if dentro del bucle con la condición de comparar cada posición de la lista del array con la que ha introducido el usuario en la variable "buscar"
  iii. Ponemos una variable contador con incremento para que recoja el valor de cada vuelta del bucle.
  iv. La siguiente sentencia if(encontrado) se encarga de dos acciones.
    1. Si en la sentencia anterior ha encontrado la igualdad, la variable "encontrado" es igual a "true" por lo que se cumple la sentencia y resta una vuelta al contador ya que dará el máximo de vueltas permitido (20) y nuestro array es de 19 posiciones por lo que daría un error.
    2. Si no ha encontrado el valor esperado significa que ha recorrido todo el array y no ha encontrado similitud por lo que la película no existe.

```
132     /**
133      * Método para buscar una película que coincida con alguna del Array lista
134      * previamente introducido, con el dato introducido por teclado.
135      *
136      *
137      * @param lista  Array lista para alamacenar peliculas.
138      * @param lisFe  Array lista para alamacenar fecha.
139      * @param lisVal Array lista para alamacenar la valoración.
140      *
141      * @return int Posición de la película buscada
142      */
143     public static int buscar(String[] lista, String[] lisFe, String[] lisVal) {
144         int longitud = lista.length;
145         int contador = 0;
146         boolean encontrado = false;
147         String buscar;
148         System.out.println("Introduce el nombre de la pelicula");
149         buscar = utilidades.pedirString();
150         do {
151             if (lista[contador].equals(buscar)) {
152                 encontrado = true;
153             }
154             contador++;
155         } while (contador < longitud && !encontrado);
156
157         /*
158          * Evitar la vuelta extra para que coincida la posicion con las listas
159          */
160         if (encontrado) {
161             contador -= 1;
162         } else {
163             System.out.println("La pelicula no existe");
164         }
165         return contador;
166     }
```

- **Método comprobacionValoracion(String texto)** → Su finalidad es comprobar que cuando el usuario introduzca una valoración se corresponda a la longitud pedida y con la simbología correcta.

  Explicación de código: El parámetro texto lanzará un mensaje por pantalla.

  i. Utilizamos un bucle do while (ya que queremos que se ejecute al menos una vez la sentencia) con la condición booleana "encontrado" OR longitud distinto de 1.
  Dentro inicializamos la variable "encontrado" igual a "false", inicializamos la variable "valoración" pidiendo la clase utilidades.pedirString() y inicializamos la longitud con el tamaño del String introducido por el usuario.

  ii. Utilizamos una sentencia if con la condición: mientras la instrucción indexOf de la variable "cadena" hacia la variable "valoracion" en la primera posición sea igual a -1, saque un mensaje de texto de "error de simbología". Ya que significa que en el dato que ha introducido el usuario no hay ninguno que se parezca a lo que contiene la variable "cadena" por lo que devuelve un -1.

  iii. Utilizamos un else if para insertar la condición de la longitud y manda un mensaje de error de longitud

```
168     /**
169      * Método de introducción por teclado de una valoracion tipo String y poste
170      * validación, controlando la simbología y la longitud.
171      *
172      * @param texto Lanza un texto por pantalla.
173      *
174      * @return String Fecha de la película en Sring.
175      */
176     public static String comprobacionValoracion(String texto) {
177         String valoracion;
178         int longitud;
179         String cadena = "1234567890";
180         boolean encontrado;
181 
182         System.out.println(texto);
183         do {
184             encontrado = false;
185             valoracion = utilidades.pedirString();
186             longitud = valoracion.length();
187 
188             if (cadena.indexOf(valoracion.charAt(0)) == -1) {
189                 encontrado = true;
190                 System.out.println("El valor introducido no es numerico vuelve
191             } else if (longitud != 1) {
192                 System.out.println("La longitud no es la correcta vuelva a intr
193             }
194         } while (encontrado || longitud != 1);
195         return valoracion;
196 
197     }
```

- **Método comprobacionFecha(String texto)** → Su finalidad es la comprobación de que los datos introducidos por el usuario en el array fecha se cumplan con los parámetros indicados.

  Explicación de código: El parámetro texto lanzará un mensaje por pantalla.

  i. Utilizamos un bucle do while (ya que queremos que se ejecute al menos una vez la sentencia) con la condición booleana "encontrado" OR longitud distinto de 4.
  Dentro inicializamos la variable "encontrado" igual a "false", inicializamos la variable "fecha" pidiendo la clase utilidades.pedirString() y inicializamos la longitud con el tamaño del String introducido por el usuario.

  ii. En este caso utilizaremos un bucle for para poder recorrer las posiciones del String fecha y realizar su comprobación una a una.

  iii. Utilizamos una sentencia if con la condición: mientras la instrucción indexOf de la variable "cadena" hacia la variable "fecha" en la primera posición sea igual a -1, saque un mensaje de texto de "error de simbología". Ya que significa que en el dato que ha introducido el usuario no hay ninguno que se parezca a lo que contiene la variable "cadena" por lo que devuelve un -1.

  iv. Utilizamos un else if para insertar la condición de la longitud y manda un mensaje de "error" de longitud

```
199    /**
200     * Método de introducción por teclado de una fecha tipo String y posterior
201     * validación, controlando la simbología y la longitud.
202     *
203     * @param texto Lanza un texto por pantalla.
204     *
205     * @return String Fecha de la película en Sring.
206     */
207    public static String comprobacionFecha(String texto) {
208        String fecha;
209        int longitud;
210        String cadena = "1234567890";
211        boolean encontrado;
212
213        System.out.println(texto);
214        do {
215            encontrado = false;
216            fecha = utilidades.pedirString();
217            longitud = fecha.length();
218
219            for (int i = 0; i < longitud && !encontrado; i++) {
220                if (cadena.indexOf(fecha.charAt(i)) == -1) {
221                    encontrado = true;
222                    System.out.println("El valor introducido no es numerico vue
223                } else if (longitud != 4) {
224                    System.out.println("La longitud no es la correcta vuelva a
225                    encontrado = true;
226                }
227            }
228        } while (encontrado || longitud != 4);
229        return fecha;
230    }
```

- **Método mostrar(String[] lista, String[] lisFe, String[] lisVal)** → Su finalidad es mostrar el contenido de los arrays

  Explicación del código: mediante un bucle for le indicamos al System.out.println("") que muestre por pantalla el mensaje concatenado que hay en su interior cogiendo los valores de los parámetros declarados.

```
232●    /**
233      * Método para mostrar los datos, previamente introducidos mediante otros
234      * métodos, y sacarlos por pantalla.
235      *
236      *
237      * @param lista  Array lista para alamacenar peliculas.
238      * @param lisFe  Array lista para alamacenar fecha.
239      * @param lisVal Array lista para alamacenar la valoración.
240      *
241      */
242●    public static void mostrar(String[] lista, String[] lisFe, String[] lisVal)
243          int longitud=lista.length;
244
245          for(int i=0;i<longitud;i++) {
246                  System.out.println("Pelicula: " + lista[i] + "; Fecha (" + lisF
247          }
248      }
```

- Método main → Su finalidad es ejecutar los métodos anteriormente mencionados haciendo una combinación de instrucciones donde se lanzará por pantalla un menú y el usuario debe elegir una de las opciones añadir, modificar, borrar, consultar, salir.

  Explicación de código:

  i. Utilización de método inicializar(datos,fecha,valoración, inicializándolos a ""); (Ahora las 20 posiciones de los arrays tienen el valor "").
  ii. Utilización de un bucle do while donde su condición sea que se repita mientras que la variable "num" , que es igual a la clase utilidades.pedirNumMenu(); sea distinto de 5.
  iii. Utilización de sentencia switch para la elección de la opción previamente introducida.
    1. Caso 1(Añadir): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
       a. Se pide al método buscarLibre(); que busque la primera posición con el contenido "".
       b. Con la sentencia if nos cercioramos de que haya hueco en el array ya que el programa devolverá un valor=-1 por lo que saldrá mensaje de "array lleno". Si hay hueco se ejecutarán las instrucción donde se guardarán , con la posición recogida anteriormente en el método buscarLibre, en los tres arrays mediante la utilización de los métodos, utilidades.pedirString(), comprobacionFecha(), comprobacionValoracion.

2. Caso 2 (Modificar): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
    a. Utilizaremos una variable "posicionComodin" para recoger el valor del método buscar();
    b. Con la sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
    c. Se ejecutarán las instrucción donde se guardarán los nuevos datos introducidos en los arrays con los métodos, utilidades.pedirString(), comprobacionFecha(), comprobacionValoracion.

3. Caso 3 (Borrar):): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
    a. Utilizaremos una variable "posicionComodin" para recoger el valor del método buscar();
    b. Con la sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
    c. En este caso se completarán las posiciones previamente buscadas con el valor ("");
4. Caso 4 (Consultar): Utilizamos un bucle do while para realizar una sentencia repetitiva que nos evitará salir al menú principal y poder añadir películas sin interrupción.
    a. Con la sentencia if else elegiremos una opción u otra
    d. Anidando otra sentencia if me cercioro de que el valor recogido por buscar sea distinto del tamaño del array ya que en ese caso daría error de consola y significaría que no ha encontrado el valor que el usuario ha introducido.
    e. Salida de mensaje concatenado con los parámetros.

```java
/**
 * Programa CRUD donde mediante interacción por teclado se navegará por las
 * distintas opciones del menú con el tema peliculas
 *
 *
 * @author Alfonso Lopez Delgado
 */
public class menuFinal {

    public static void main(String[] args) {
        int num, posicionComodin;
        int posicionContador = 0;
        String[] datos = new String[20];
        String[] fecha = new String[20];
        String[] valoracion = new String[20];

        inicializar(datos, fecha, valoracion, "");
        do {
            menu();
            num = utilidades.pedirNumMenu();
            switch (num) {
            case 1:
                do {
                    posicionComodin = buscarLibre(datos, fecha, valoracion, "")
                    if (posicionComodin != -1) {
                        System.out.println("Introduce una pelicula");
                        datos[posicionComodin] = utilidades.pedirString();
                        fecha[posicionComodin] = comprobacionFecha("Introduce u
                        valoracion[posicionComodin] = comprobacionValoracion("I
                    } else {
                        System.out.println("Ha superado el maximo de peliculas
                    }

                } while (utilidades.confirmacion("Desea seguir creando pelicula
                break;
            case 2:
                do {
                    posicionComodin = buscar(datos, fecha, valoracion);
                    if (posicionComodin != datos.length) {
                        System.out.println("Introduce el nombre de una nueva pe
                        datos[posicionComodin] = utilidades.pedirString();
                        fecha[posicionComodin] = comprobacionFecha("Introduce l
                        valoracion[posicionComodin] = comprobacionValoracion(
                                "Introduce la valoracion de la nueva pelicula")
                    }
                } while (utilidades.confirmacion("Desea modificar otra pelicula

                break;
            case 3:
                do {
                    posicionComodin = buscar(datos, fecha, valoracion);
                    if (posicionComodin != datos.length) {
                        System.out.println("La pelicula se ha borrado correctam
                        datos[posicionComodin] = "";
                        fecha[posicionComodin] = "";
                        valoracion[posicionComodin] = "";
                    }
                } while (utilidades.confirmacion("Desea borrar otra pelicula (s

                break;
            case 4:
                do {
                    if (utilidades.confirmacion("Quiere mostrar todo (s) o busc
                        mostrar(datos, fecha, valoracion);
                    } else {
                        posicionComodin = buscar(datos, fecha, valoracion);
                        if (posicionComodin != datos.length) {
                            System.out.println("La pelicula " + datos[posicionC
                                    + fecha[posicionComodin] + ". Tiene una val
                                    + valoracion[posicionComodin] + "/10)");
                        }
                    }
                } while (utilidades.confirmacion("Desea hacer otra consulta (s/
                break;
            default:
                System.out.println("Gracias por utilizar el Menu de MiniMestri
            }
        } while (num != 5);
    }

```

GIT

El programa GitHub es bastante útil para trabajar en equipo, la detección de fallos sin tocar el programa principal o actuar como un manual histórico de los pasos que hemos hecho.

En primer lugar, he creado una rama principal donde guardaré todos mis archivos que contendrán el proyecto final y en paralelo he creado una rama nueva para trabajar sobre ella para evitar errores o fallos indeseados que puedan estropear el proyecto principal.

Mientras trabajaba con esta segunda rama he podido especificar cada cambio que hacia en el proyecto mediante comentarios o he descartado los antiguos.

Después de cada jornada actualizaba los cambios en la nube.

Una vez finalizado todas las partes del proyecto fusioné la rama secundaria con la principal para ver una única rama.

https://github.com/MiniMestri/CRUDArrays.git