



Escuela Politécnica Nacional

Facultad De Ingeniería en Sistemas

PROGRAMACIÓN II



PERÍODO ACADÉMICO: 2022B

ASIGNATURA: IICCD244

PROFESOR: Maritzol R. Tenemaza V.

TIPO DE INSTRUMENTO: Proyecto Final

FECHA DE ENTREGA: 05-03-2023

NOMBRE ESTUDIANTES:

- Garces P. Boris G
- Oñate G. Ian A.
- Robalino T. Valeria C.
- Singaña T. Josune A.
- Yar S. Kenneth J.

CARRERA: COMPUTACIÓN (RRA20)

ÍNDICE

Contenido

PROYECTO IB-Biblioteca	2
OBJETIVO	2
PROCESO.....	2
MYSQL(BASE DE DATOS) Y CONEXIÓN CON NETBEANS.....	2
Instalación MySQL	2
Creación de Base de datos y tablas en MySQL.....	12
Generar las tablas y sus relaciones referentes.....	12
Crear la ventana login, si ingresa, conectar a la base de datos.....	14
PAQUETES Y CLASES EN EL PROYECTO BIBBLIOTECA.....	23
Paquete Negocio.....	23
Paquete Visual	26
RESULTADOS	58
CONCLUSIONES Y RECOMENDACIONES	78
BIBLIOGRAFÍA	79

PROYECTO IB-Biblioteca

OBJETIVO

- Diseñar una base de datos con tablas que contengan la información de estudiantes, libros y préstamos respectivamente.
- Crear un programa en Netbeans que permita realizar registro de estudiantes, libros y realizar préstamos y devoluciones de textos.

PROCESO

MYSQL(BASE DE DATOS) Y CONEXIÓN CON NETBEANS

Instalación MySQL

En primer lugar, dentro de la página oficial de MySQL, se selecciona la opción MySQL Community. <https://dev.mysql.com/downloads/mysql/>

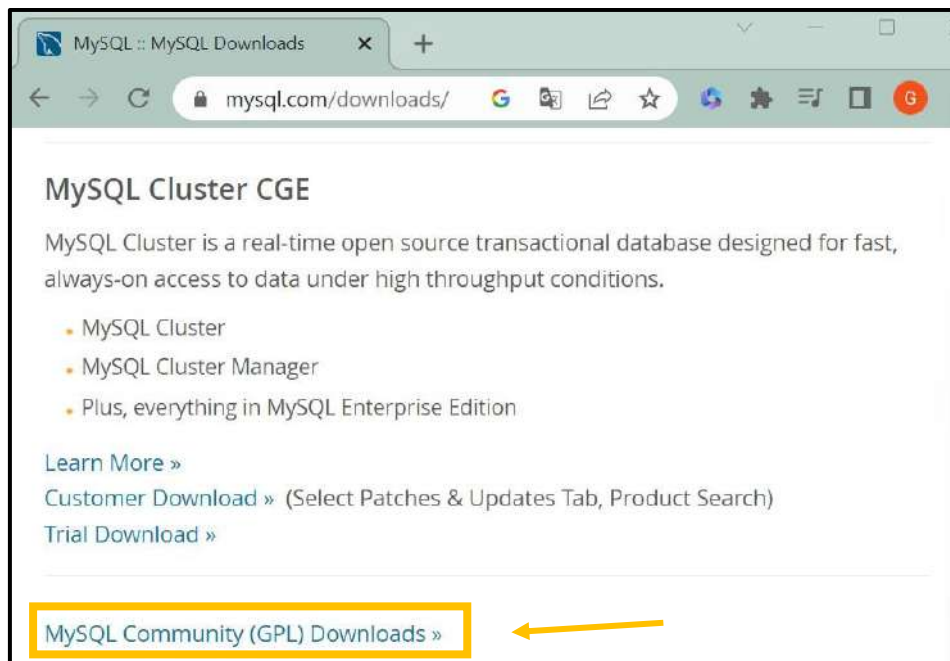


Figura1.- Captura página oficial de MySQL. Después, se despliega una ventana, en la cual se visualiza un listado donde se elige la opción MySQL Community Server.



Figura2.- Captura despliegue de opciones MySQL Community Downloads. A continuación, de acuerdo con el sistema del dispositivo que se pretende realizar la instalación, se da clic en Go to Download Page.

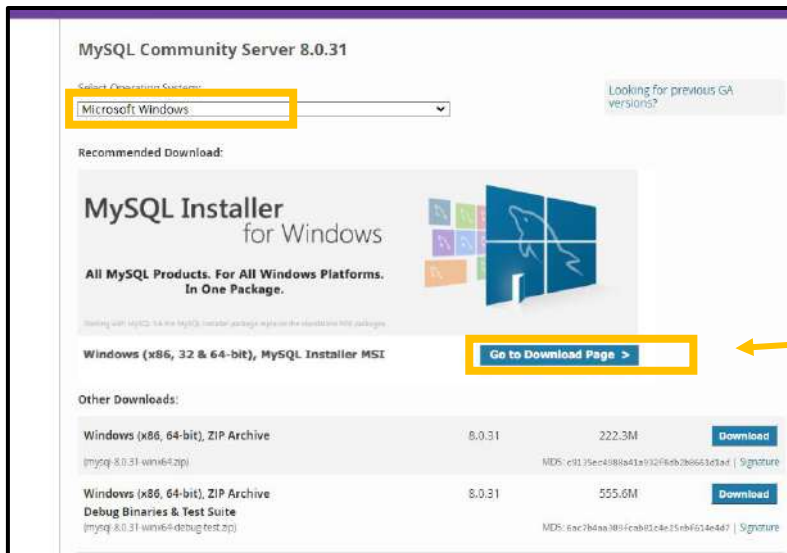
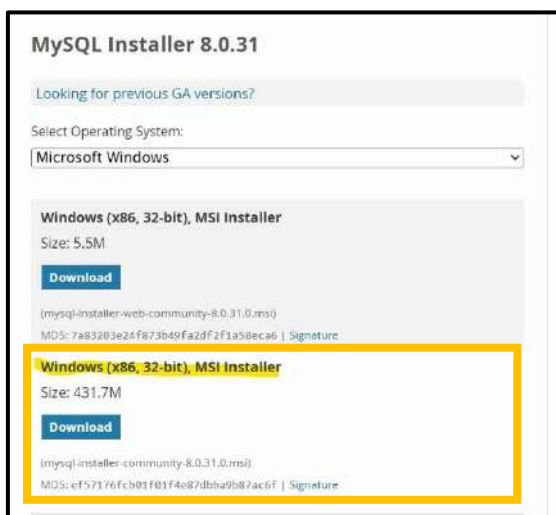


Figura2.- Captura despliegue de opciones MySQL Community Downloads. En Go to Download Page se abre una pantalla que brinda la oportunidad de elegir entre un paquete con todo incluido y otro que ira descargando librerías a medida que se utilice. Se escoge el paquete con todo incluido.



Nota: MySQL Installer es 32 bit, pero se puede instalar en 32 bit y 64 bit binaries.

Figura3.- Captura opciones para instalar MySQL Installer 8.0.31.

Tanto para x32, x64 u otro tipo de sistema operativo, la ventana a continuación es la mismo y se puede crear una cuenta Oracle, o en este caso se opta por la opción **No thanks, just start my download.**

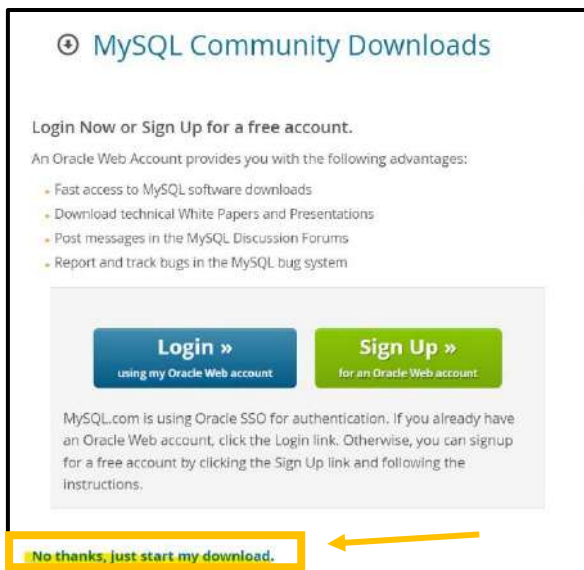



Figura4.- Captura paso previo instalar paquete.

Se procede a abrir el instalador descargado:  mysql-installer-community-8.0.31.0

Posteriormente, aparece una ventana que da la posibilidad de elegir el tipo de Setup que se desee instalar, dado nuestro caso de un servidor MySQL podemos elegir entre Developer Default, Server only y Full.

Develop Default: Instala el servidor MySQL y las herramientas necesarias para el desarrollo de aplicaciones MySQL. Esto es útil si pretende desarrollar aplicaciones para un servidor existente.

Este tipo de instalación incluye:

- * Servidor MySQL

El sistema de gestión de bases de datos SQL de código abierto más popular.

- * Consola MySQL

La nueva aplicación de cliente MySQL para administrar servidores MySQL e instancias de clúster de InnoDB.

- * Enrutador MySQL

Daemon de enrutador de alta disponibilidad para configuraciones de clúster InnoDB que se instalarán en los nodos de la aplicación.

- * Banco de trabajo MySQL

La aplicación GUI para desarrollar y administrar el servidor.

- * MySQL para VisualStudio

Para trabajar con el servidor MySQL de VS.

- * Conectores MySQL

Connector/Net, Java, C/C++, ODBC y otros

- * Ejemplos y tutoriales

Para ayudarle a empezar con su desarrollo.

* Documentación

Le permite leer la documentación sin conexión.

Server only: Instala solo el servidor MySQL. Este tipo debe usarse donde desee implementar un servidor MySQL, pero no desarrollará aplicaciones MySQL.

Full: Instala todos los productos disponibles en este catálogo, incluidos MySQL Server, MySQL Shell, MySQL Router, MySQL Workbench, MySQL Connectors, documentación, ejemplos y mucho más.

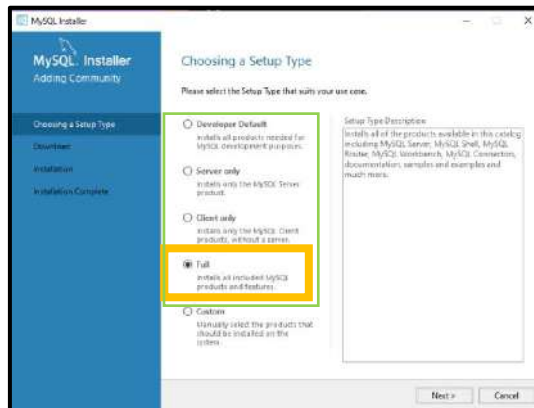


Figura5.- Captura elección Setup Type(full).

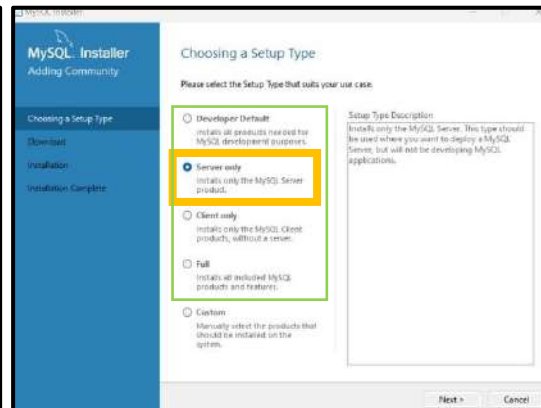


Figura6.- Captura elección Setup Type(server only).

A continuación, se muestra una ventana con los requerimientos para el tipo de instalación que se elige, en el caso que falte algún requisito se instalará automáticamente después de dar clic en Execute, en algunos casos se requiere instalación manual.

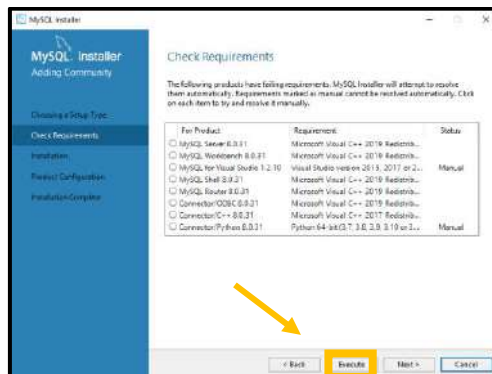


Figura7.- Captura requisitos Setup Type(full).

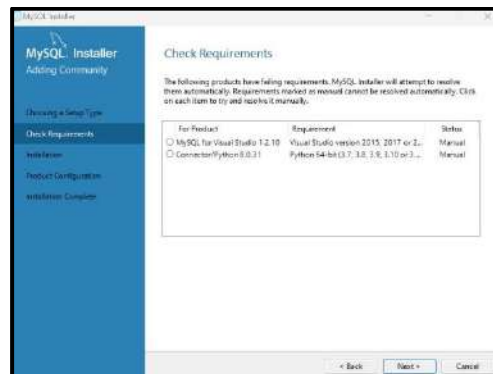


Figura8.- Captura requisitos Setup Type(server only).

Ejemplos de instalaciones manuales necesarias: (Microsoft Visual, Visual studio, Phytion)



Cuando se tiene todos los requisitos tanto manuales como los del instalador, los ítems se colorean de verde:

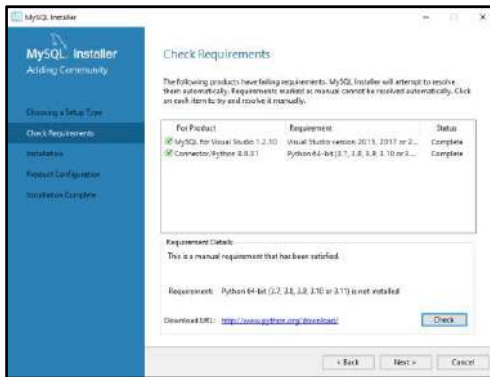


Figura9.- Captura requisitos completos Setup Type(full).

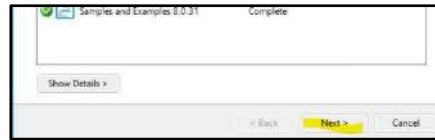


Figura10.- Captura requisitos completos Setup Type(server only).

Luego de completar con los requisitos se selecciona next. Enseguida aparece un listado de los productos a instalar. Se selecciona execute para proceder a la instalación de estos.

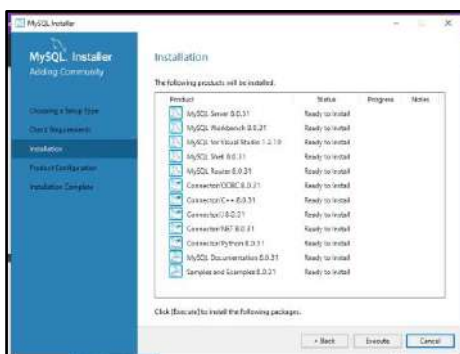


Figura11.- Captura listado productos Setup Type(full).

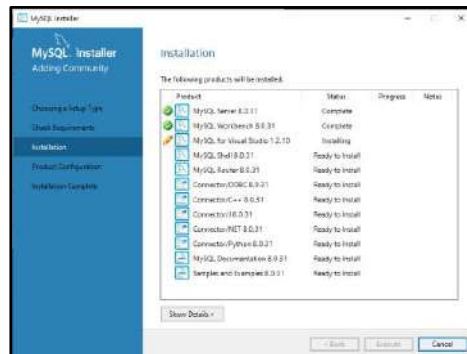


Figura12.- Captura instalación productos Setup Type(full).

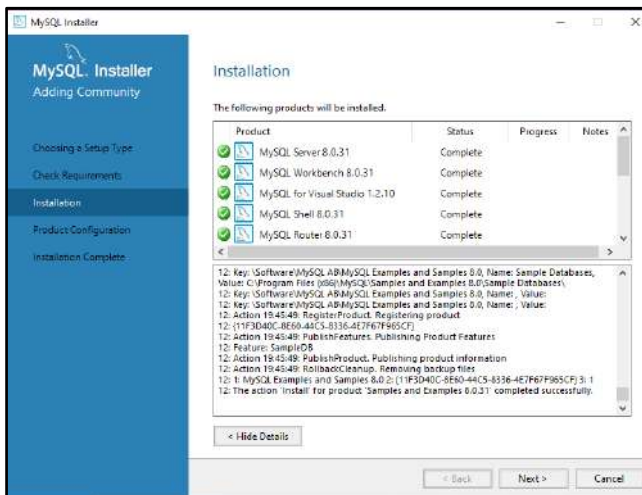


Figura12.- Captura instalación completa de herramientas Setup Type(full). En la pantalla de configuración de productos, se puede ver un listado de los productos que necesitan ser configurados, dando clic en next se pasa a la pantalla para la configuración del MySQL Server.

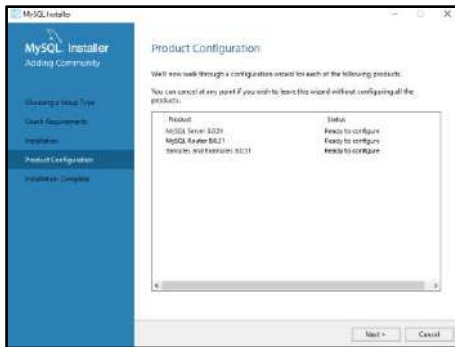


Figura13.- Captura listado productos a configurar Setup Type(full).

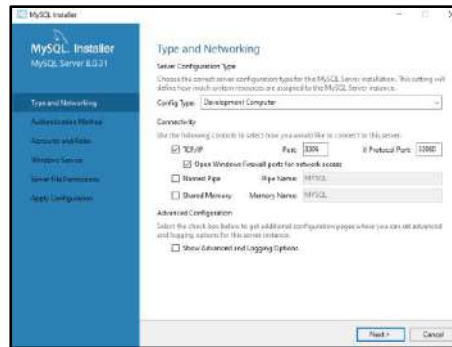


Figura14.- Captura configuración del MySQL Server Setup Type(full).

El tipo de configuración de MySQL (recuadro amarillo) es un conjunto predefinido de parámetros de configuración que determina cuántos recursos se deben asignar a los servicios de MySQL. Tienes tres opciones de configuración:

Computadora de desarrollo: esta configuración utiliza una cantidad mínima de recursos para el servicio MySQL.

Equipo servidor: esta configuración utiliza una cantidad mínima de recursos. Esta opción es adecuada cuando estamos instalando servidores de base de datos y servidores web en la misma máquina. La configuración asigna una cantidad promedio de recursos al servicio MySQL.

Equipo Dedicado: Esta opción se utiliza cuando hemos creado un Servidor MySQL dedicado. La configuración asigna una gran cantidad de recursos al servicio MySQL.

Conectividad de red (recuadro morado).

En esta sección, podemos controlar cómo los clientes pueden conectarse a las bases de datos MySQL. Podemos usar el protocolo TCP/IP o Named Pipe o Shared Memory. Si desea configurar Canalización con nombre/Memoria compartida, debemos proporcionar el Nombre de canalización y el Nombre de memoria. También puede especificar el puerto predeterminado para conectarse al servidor de la base de datos. También puede optar por permitir el número de puerto especificado en el cuadro de texto Puerto en el cortafuegos.

Por ahora lo dejaremos en por defecto.

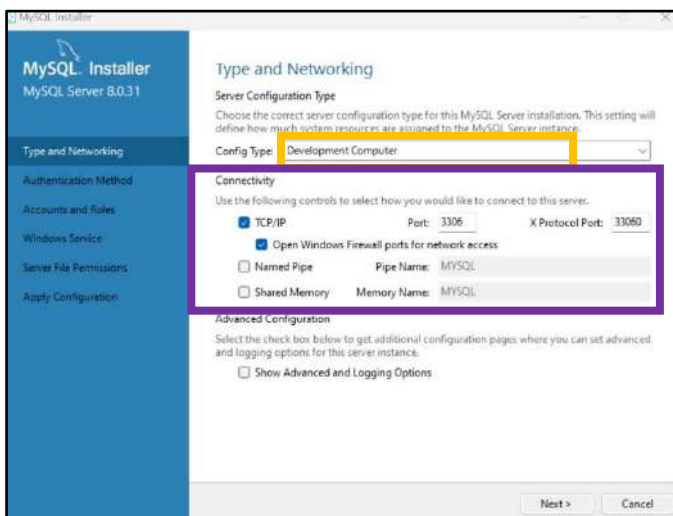


Figura15.- Captura listado productos a configurar Setup Type(server only). Después, se abre la pantalla para configurar el tipo de autenticación se utilizará, recomendación dejarlo por defecto.

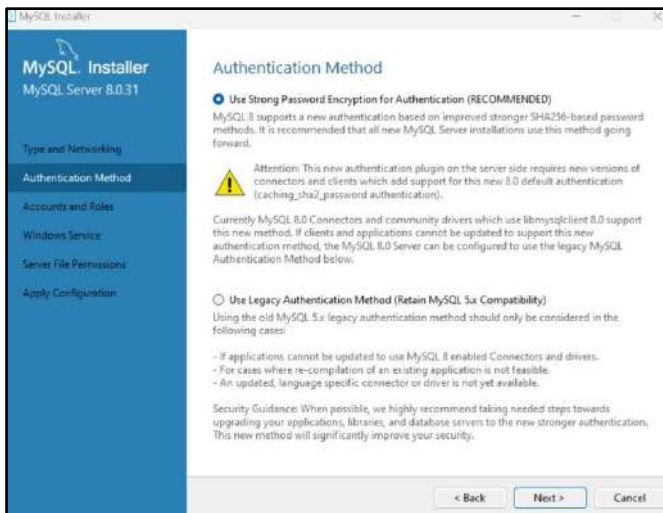


Figura16.- Captura pantalla de método de autenticación. En la pantalla de roles y cuentas (Accounts and Roles), se especifica the MySQL root account password (contraseña de la cuenta raíz de MySQL). EN la parte inferior se pueden crear usuario con contraseñas.

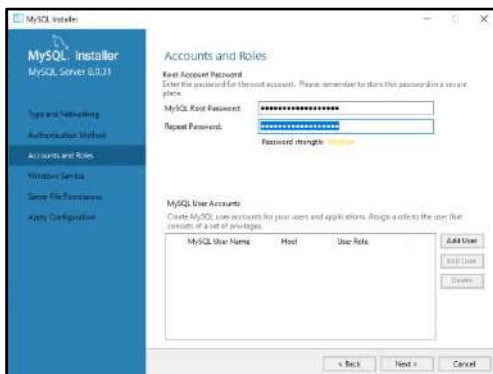


Figura17.- Captura pantalla de cuentas y roles. Para la siguiente pantalla establecer algunas configuraciones de MySQL server para que ejecute con un servicio de Windows, sin embargo, se recomienda utilizar las opciones por defecto.

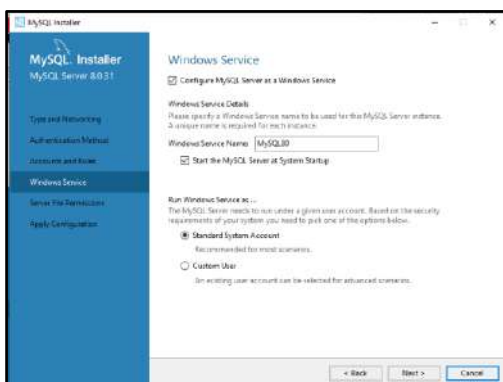


Figura18.- Captura pantalla de servicio Windows. En la siguiente ventana se establecen los permisos que tendrá el servidor, se recomienda otorgar un acceso completo y seleccionar next.

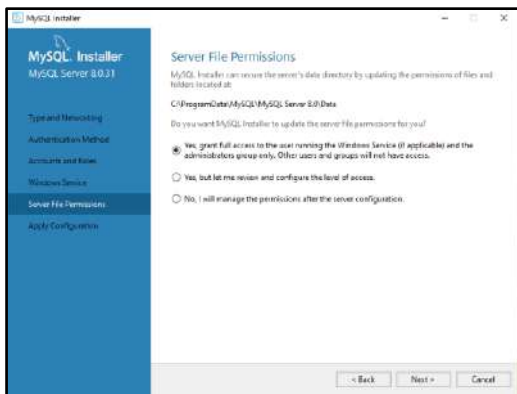


Figura19.- Captura permisos acceso carpeta servidor.

Luego, en la pantalla Aplicar configuración, puede ver la lista de pasos de confirmación.

Una vez que se verifiquen todos los ajustes de configuración, hacer clic en Ejecutar.

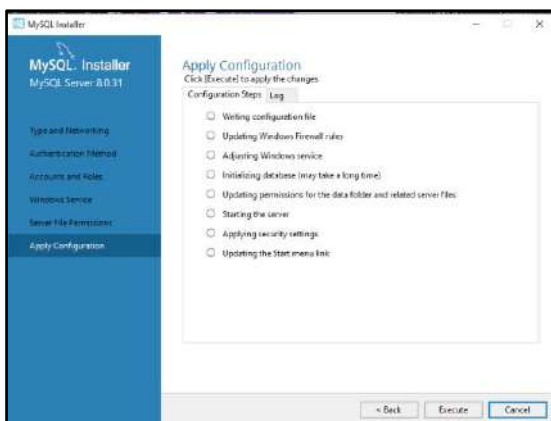


Figura20.- Captura Apply Configuration

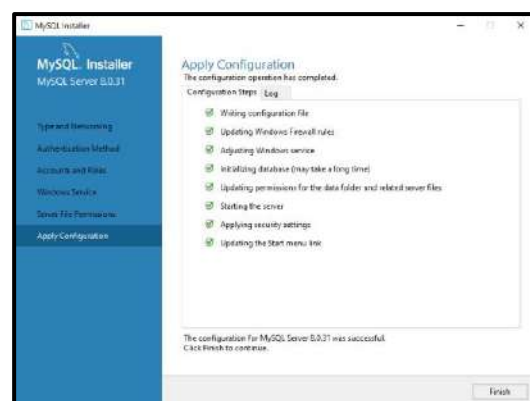
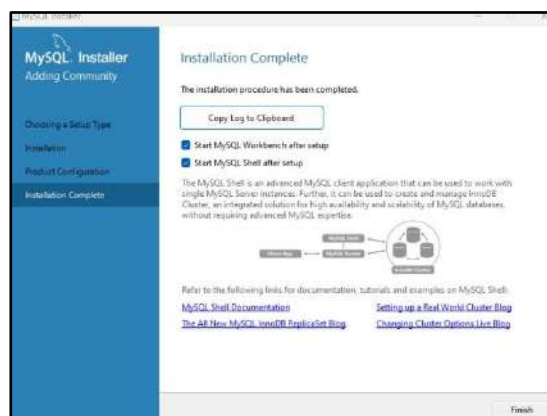
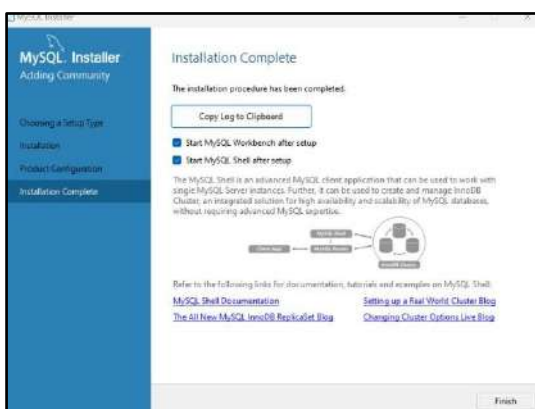


Figura21.- Captura configuración propiedades programa.

Una vez que todo este instalado, se presiona en Finish.

Si se eligió Server only las pantallas que se presentan son: Seguidamente la instalación se completará.



Instalar la base de datos de muestra

Si ha elegido instalar todos los componentes de MySQL Server (Tipo de instalación completa), el instalador de MySQL se mueve a la pantalla Muestra y Ejemplo. En esta pantalla, proporcione el nombre de usuario y la contraseña del usuario que tiene privilegios de root/sysadmin y haga clic en Verificar. Si la conexión se establece con éxito, haga clic en siguiente.

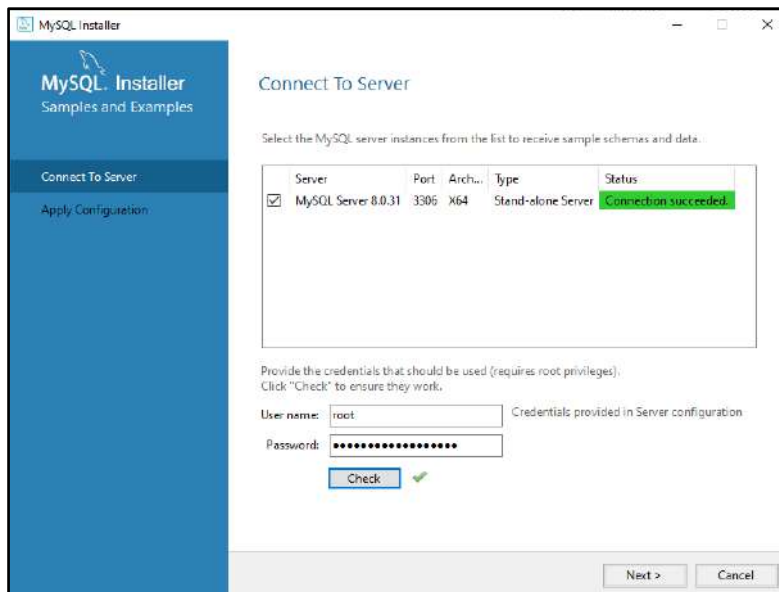


Figura22.- Captura conectar con el servidor.
En la pantalla Aplicar configuración, haga clic en Ejecutar para iniciar la instalación de la base de datos de muestra. Una vez, que la base de datos sea instalada, haga clic en terminar.

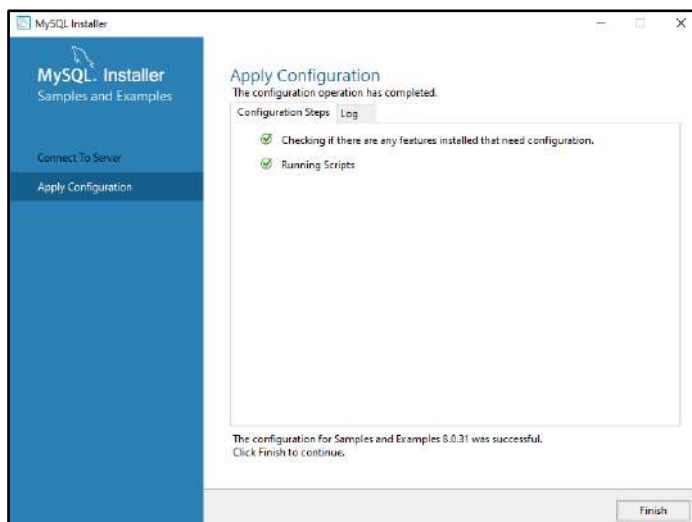


Figura23.- Captura Apply Configuration.

El instalador continúa con la pantalla Configuración del producto. En esta pantalla, puede ver que la instalación de MySQL Server 8.0.19 y Sample and Example 8.0.19 se completó con éxito.

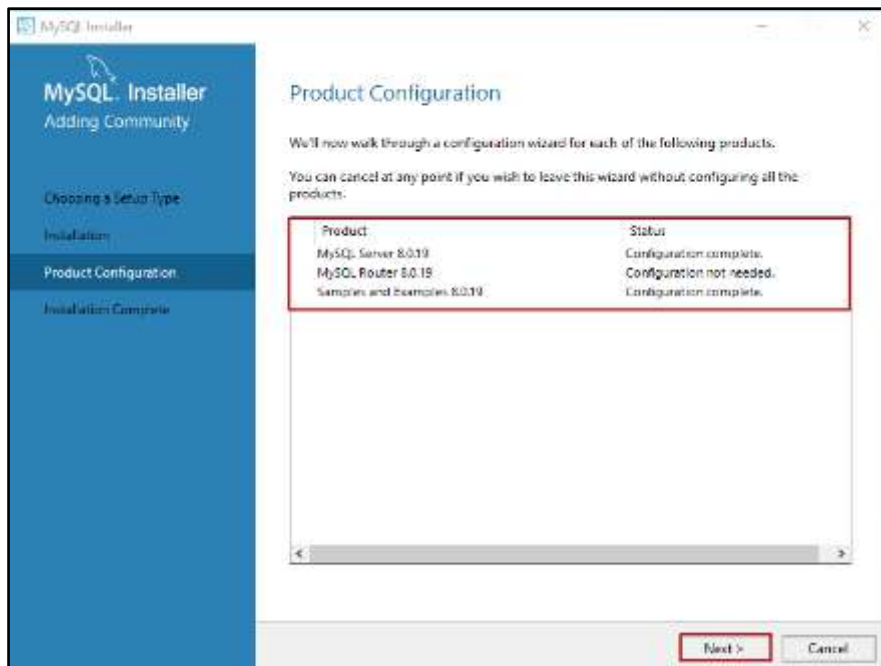


Figura24.- Captura Configuración del producto.
Una vez que se completa la instalación, puede copiar los registros de instalación en el portapapeles para revisarlos más tarde. Además, si desea comenzar a explorar MySQL de inmediato, puede seleccionar "Iniciar MySQL Workbench después de la instalación" e "Iniciar MySQL Shell después de la instalación" y hacer clic en Finalizar.

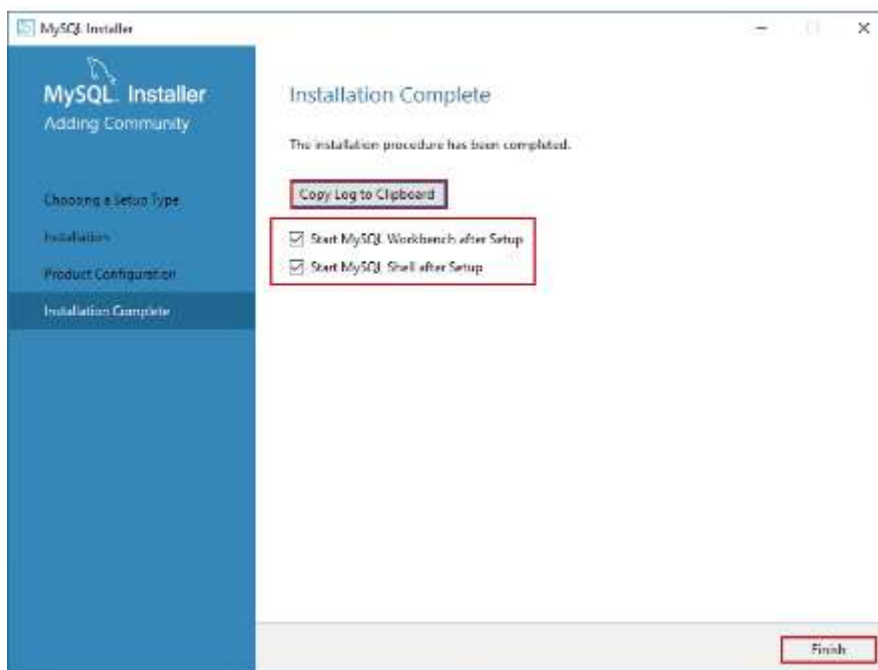


Figura25.- Captura Instalación completa.

Al hacer clic en la conexión, debe ingresar las credenciales para conectarse al servidor de la base de datos. Introduzca la contraseña y haga clic en Aceptar.

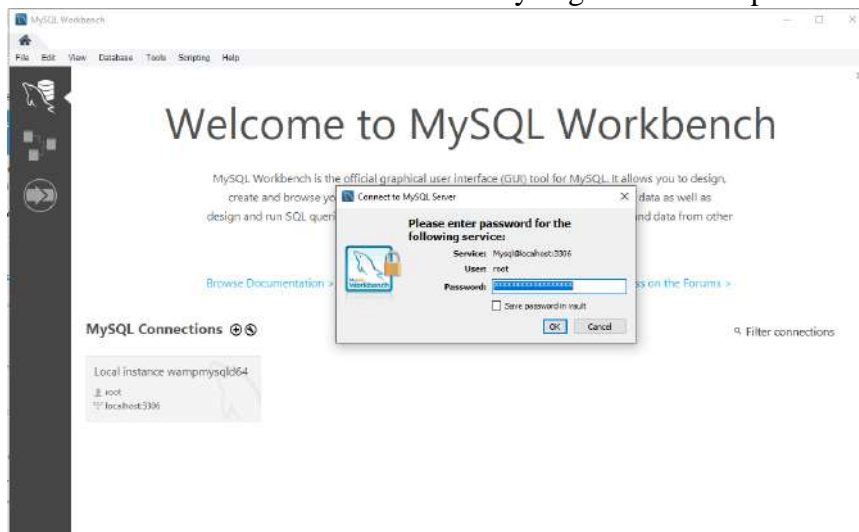


Figura 26.- Captura ingreso credenciales

Creación de Base de datos y tablas en MySQL

Una base de datos es un conjunto de datos los cuales pertenecen a un mismo contexto y son almacenados sistemáticamente para usarlos posteriormente, en otras palabras una base de datos es un “ALMACEN” donde se guardan grandes cantidades de información de una manera organizada para que después podamos encontrarla y utilizarla.

Ingresamos al workbench (usuario y clave necesario) de Mysql se coloca las siguientes líneas de código:

```
#Eliminamos la base de datos si esta creada
DROP DATABASE IF EXISTS base_libros;

#Creamos la base de datos a utilizar
CREATE DATABASE IF NOT EXISTS base_libros;
```

Generar las tablas y sus relaciones referentes

Para almacenar la información en una base de datos, la estructura que se utiliza son las tablas, en estas tablas, que adoptan el nombre del objeto que se almacena la información, se almacena la información en tuplas o registros que son una fila de la tabla correspondiente.

Es importante especificar sobre que base de datos se va a trabajar:

```
#Indicamos que base de datos vamos a utilizar de ahora en adelante
USE base_libros;
```

Entonces para registrar información en MySQL necesitamos crear tablas. Para este caso las tablas serán estudiantes, libros, estudiante_libros con sus respectivos atributos.

```
#Creamos la tabla estudiantes
CREATE TABLE IF NOT EXISTS estudiantes(
  codigo_e INT NOT NULL,
  nombre_e VARCHAR(45) NOT NULL,
  apellido_e VARCHAR(45) NOT NULL,
  fechaNacimiento_e VARCHAR(45) NOT NULL,
  cedula_e VARCHAR(10) NOT NULL,
  direccion VARCHAR(65) NOT NULL,
  telefono VARCHAR(10) NOT NULL,
  PRIMARY KEY(codigo_e)
)ENGINE = InnoDB;

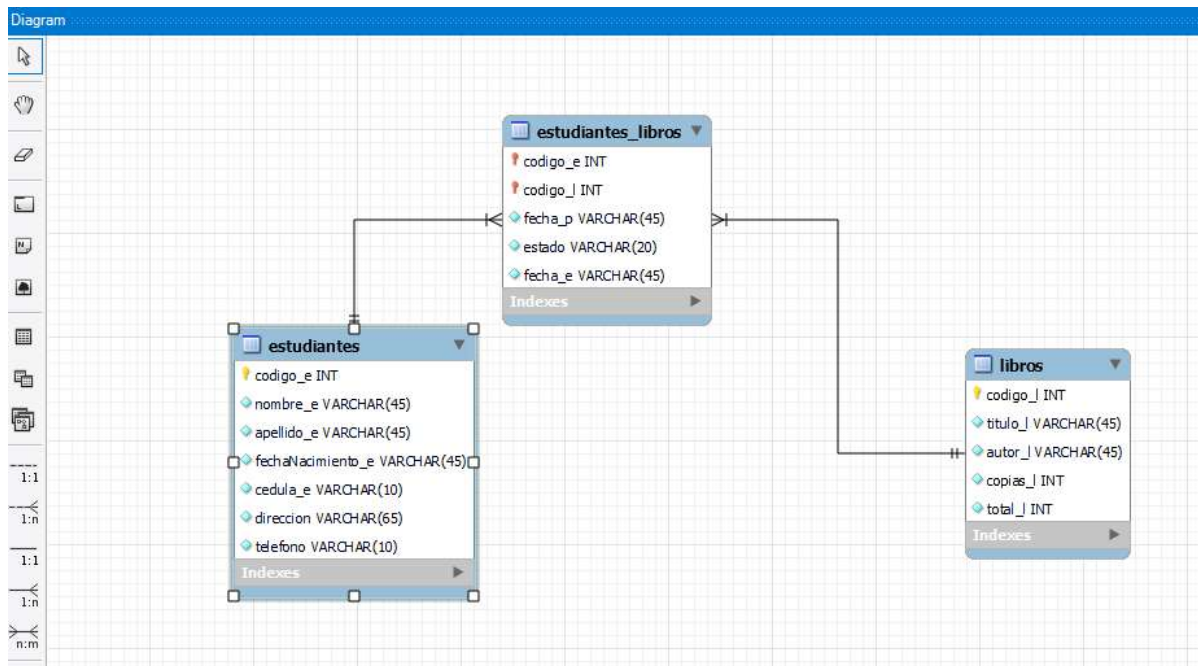
#Creamos tabla libros
CREATE TABLE IF NOT EXISTS libros(
  codigo_l INT NOT NULL,
  titulo_l VARCHAR(45) NOT NULL,
  autor_l VARCHAR(45) NOT NULL,
  copias_l INT NOT NULL,
  total_l INT NOT NULL,
  PRIMARY KEY(codigo_l)
)ENGINE = InnoDB;
```

La PRIMARY KEY es el atributo que permite diferenciar a cada tupla de una tabla, esta clave debe ser única para cada tupla de una tabla.

Para relacionar dos tablas en MySQL es necesario tener un atributo de una tabla en otra, un atributo que permita “movernos” entre la tabla y de esa forma obtener información de ese registro. Es común que el atributo escogido sea la PRIMARY KEY de la tabla que se desea relacionar este atributo se escoge ya que es la que distingue a cada registro, este atributo en la nueva tabla toma la característica de FOREIGN KEY como un atributo más de la tabla.

```
#Creamos la tabla estudiantes_libros
CREATE TABLE IF NOT EXISTS estudiantes_libros(
  codigo_e INT NOT NULL,
  codigo_l INT NOT NULL,
  fecha_p VARCHAR(45) NOT NULL,
  estado VARCHAR(20) NOT NULL,
  fecha_e VARCHAR(45) NOT NULL,
  PRIMARY KEY(codigo_l, codigo_e),
  FOREIGN KEY (codigo_l)
  REFERENCES libros(codigo_l),
  FOREIGN KEY (codigo_e)
  REFERENCES estudiantes(codigo_e)
)ENGINE = InnoDB;
```

Recordemos que la relación es un estudiante puede tener varios libros y un libros puede ser alquilado por muchos estudiantes (relación muchos a muchos). El siguiente diagrama representa dicha relación:



MySQL permite desde el espacio de trabajo (Workbench) ingresar datos de la siguiente forma:

#Insertamos los registros de estudiantes

```
INSERT INTO estudiantes VALUES(20210071, "Juan Alberto", "Castillo Salcedo", "2/12/2003", "0504111501", "M");
INSERT INTO estudiantes VALUES(20209875, "Carla Maria", "Velasgui Lopez", "7/1/2000", "0956512334", "Inc");
INSERT INTO estudiantes VALUES(20209842, "Fernando Mateo", "Perez Enriquez", "12/11/2002", "1594114523", "Car");
INSERT INTO estudiantes VALUES(20184698, "Cristian José", "Gozales Lasso", "27/3/1999", "1766653422", "Car");
INSERT INTO estudiantes VALUES(20190784, "Fernanda Soledad", "Benalcazar López", "19/5/1998", "0204999432", "Car");
INSERT INTO estudiantes VALUES(20211659, "Sandra Maria", "Ramírez Paucar", "15/10/2000", "0522141599", "Fl");
```

#Insertamos los registros de libros

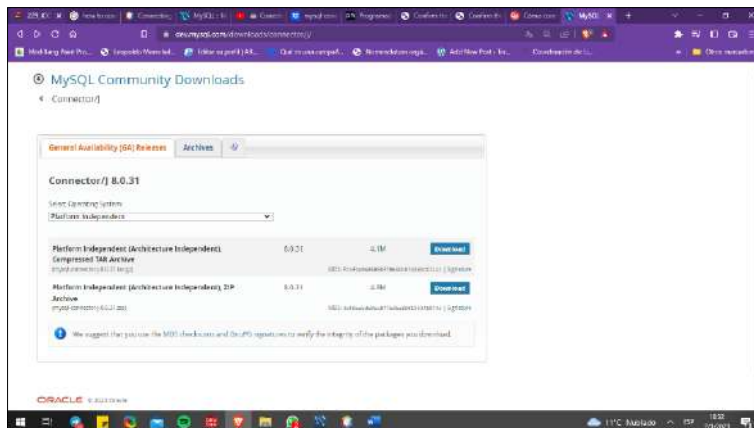
```
INSERT INTO libros VALUES(100032, "Cien años de soledad", "Gabriel García Márquez", 19,20);
INSERT INTO libros VALUES(13100876, "1984", "George Orwell", 12,12);
INSERT INTO libros VALUES(101201, "Orgullo y prejuicio", "Jane Austen", 8,8);
INSERT INTO libros VALUES(200987, "El retrato de Dorian Gray", "Oscar Wilde", 3,3);
INSERT INTO estudiantes_libros VALUES(20210071, 100032, "27/2/2023", "No Entregado", "sin fecha");
```

Crear la ventana login, si ingresa, conectar a la base de datos.

Para establecer una conexión es necesario un componente desarrollado por el equipo de MySQL denominado Connector/J. Se puede acceder mediante:

<https://dev.mysql.com/downloads/connector/j/>

Donde se elige la opción platform Independent, y presiona el botón Download una vez que se elija el tipo de archivo.



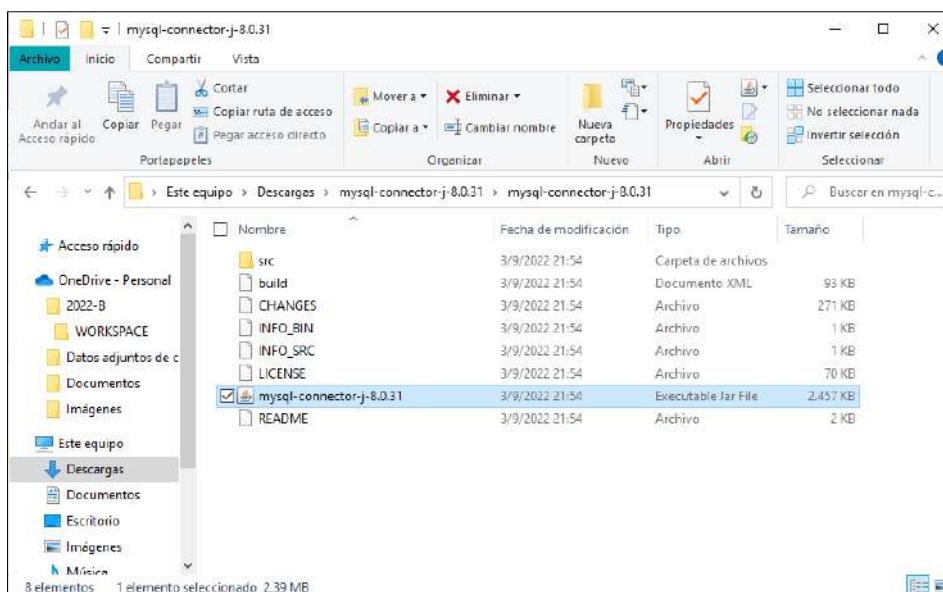
Captura página oficial de MySQL Community Download.

En la siguiente pagina solicita iniciar sesión, por lo que se utiliza No thanks, just start my download.



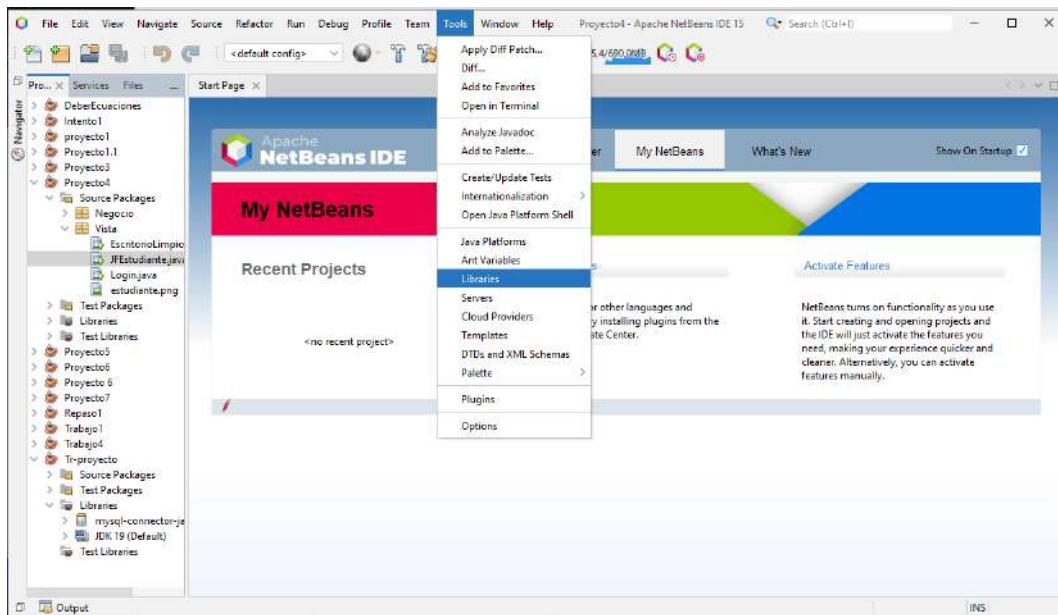
Captura solicitud iniciar sesión con una cuenta Oracle.

A continuación de la descarga, se extrae el contenido del archivo. Dentro se halla un archivo con extensión JAR, que se llama **mysql-connector-java**.



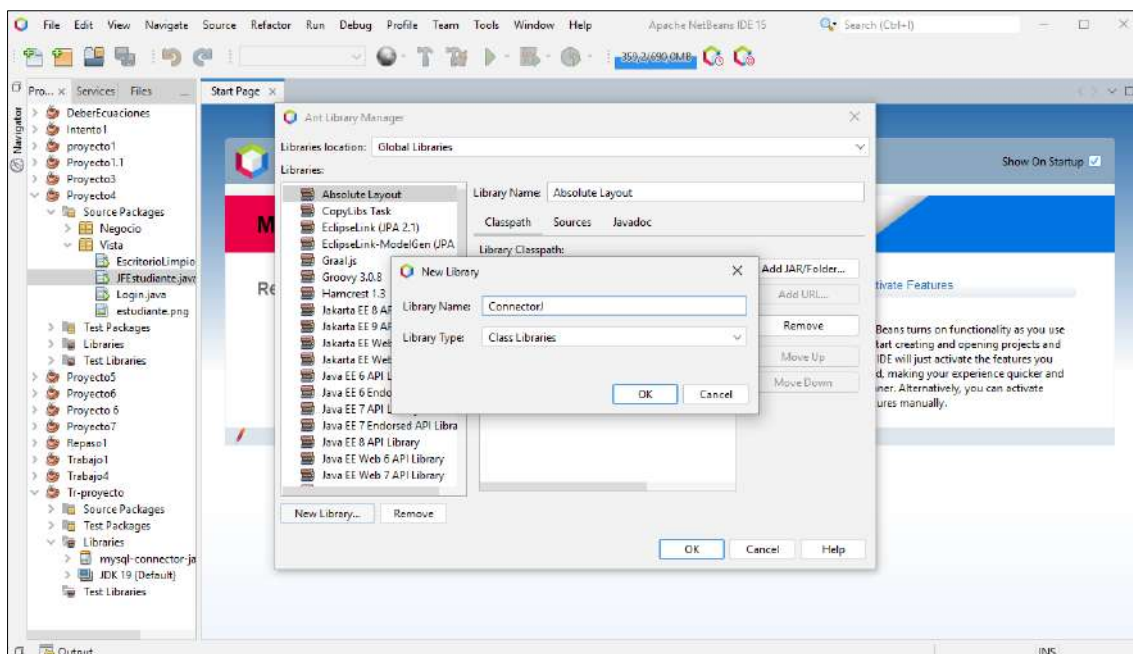
Captura archivos de mysql-connector-j-8.0.31

Después de conseguir el archivo **mysql-connector-java**, es hora de configurar netbeans, como primer paso se dirige al menú tools y selecciona Libraries.

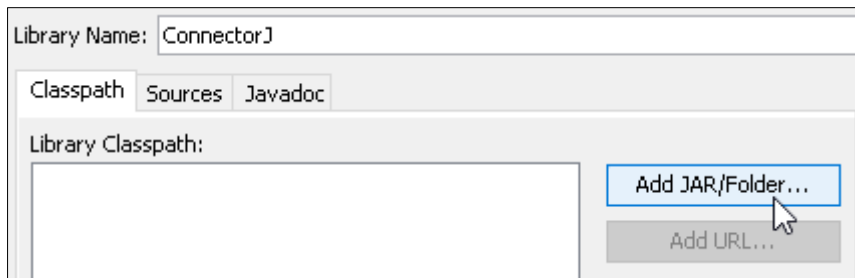


Captura menú tools opción Libraries.

En el Ant Library Manager, se da clic en el botón New Library que se encuentra en la parte inferior de la ventana. En la nueva ventana se establece el nombre de la librería al finalizar se presiona OK.

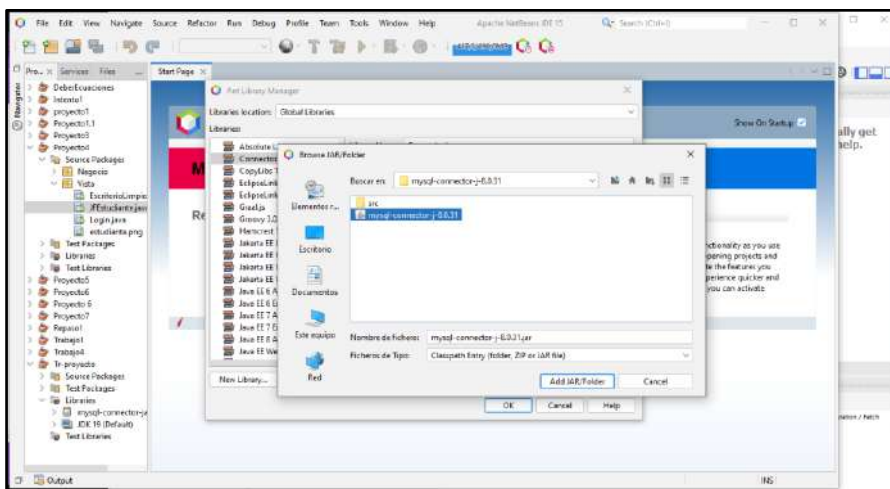


Captura creación librería en Netbeans.
A continuación, se presiona el botón Add JAR/Folder.

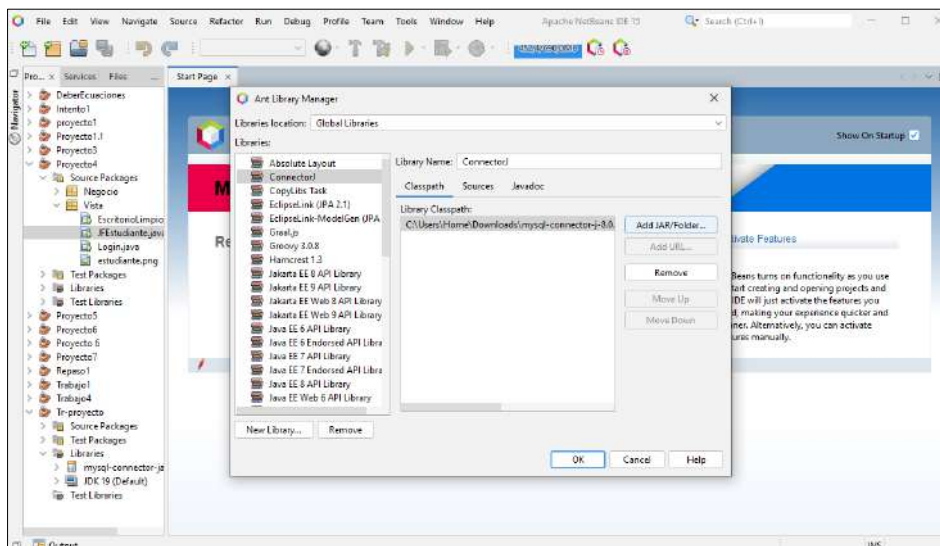


Captura botón Add JAR/Folder.

Se localiza el archivo mysql-connector-java que se descargó al principio. Al encontrarlo se selecciona y presiona el botón Add JAR/Folder.



Finalmente presiona el botón OK para cerrar el Ant Library Manager.

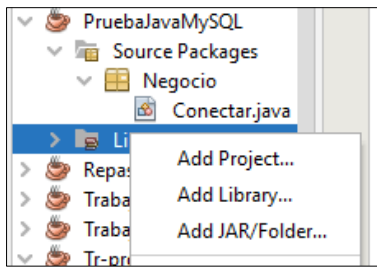


Captura Ant Library Manager.

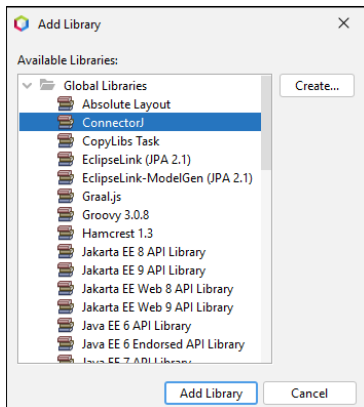
Se crea un nuevo proyecto (Java with Ant, Java Application) con el nombre Biblioteca.

Después, se crea un nuevo paquete con el nombre BaseDatos, y dentro coloca una clase con el nombre Conectar.

En la carpeta Libraries, se presiona el botón derecho del ratón y seleccionamos Add Library.



En Available Libraries, seleccionamos nuestra librería y presionamos el botón Add Library.



Para el código de la clase se añaden los siguiente imports:

```
import java.sql.Connection;
import java.sql.DriverManager;
```

Luego, se declara dentro de la clase la constante URL que incluirá los siguientes valores:

```
public static final String URL = "jdbc:mysql://localhost:3306/proyecto";
```

Como podemos observar, nos conectamos usando el nombre de dominio y puerto del servidor de base de datos. El último valor (proyecto en nuestro ejemplo) es el nombre de la base de datos.

Se añade el usuario y la clave.

```
public static final String USER = "root";
public static final String CLAVE = "";
```

A continuación, se crea un método que regresa un valor de tipo Connection.

```
public Connection getConexion(){
```

Se instancia el objeto Connection.

```
Connection con = null;
```

Después, dentro de un try, llamamos a la clase que maneja el Driver para la base de datos. También haremos la conexión pasando las tres constantes iniciales.

```
try{
    Class.forName("com.mysql.cj.jdbc.Driver");
    con = (Connection) DriverManager.getConnection(URL, USER, CLAVE);
}
```

En el catch, mostramos cualquier posible error de conexión

```
catch(Exception e){
```

```

        System.out.println("Error: " + e.getMessage());
    }
}

```

Finalmente, devolver la variable de tipo Connection y cerrar el método.

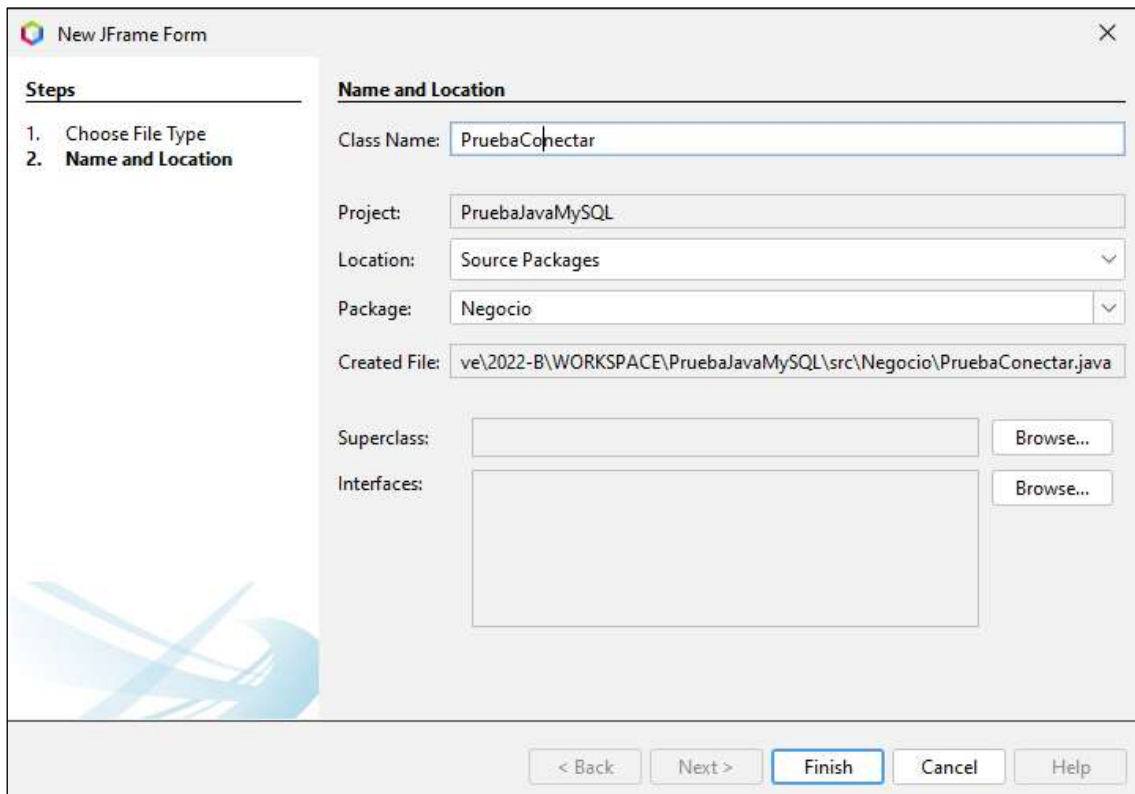
```

return con;
}

```

En el servidor de base de datos, creamos una base de datos con el mismo nombre que usamos en la clase Conectar.

Crear JFrame.



Una vez aparezca el formulario, vamos a la sección Palette (usualmente a la derecha de la pantalla) y en Swing Controls seleccionamos Button.

Se instancia la clase conectar:

```

Conectar conecta = new Conectar();

```

Después, creamos una variable de tipo Connection que reciba la conexión creada por nuestro método getConexion de la clase Conectar.

```

Connection con = conecta.getConexion();

```

Finalmente, con un JOptionPane, mostramos un mensaje indicando que hemos logrado la conexión.

```

JOptionPane.showMessageDialog(null, "Conexión establecida con éxito");

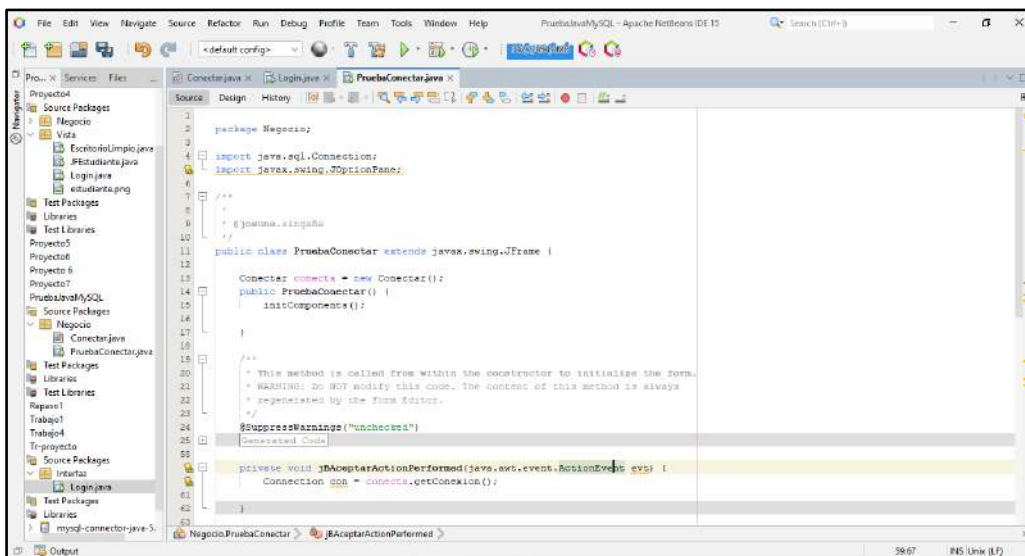
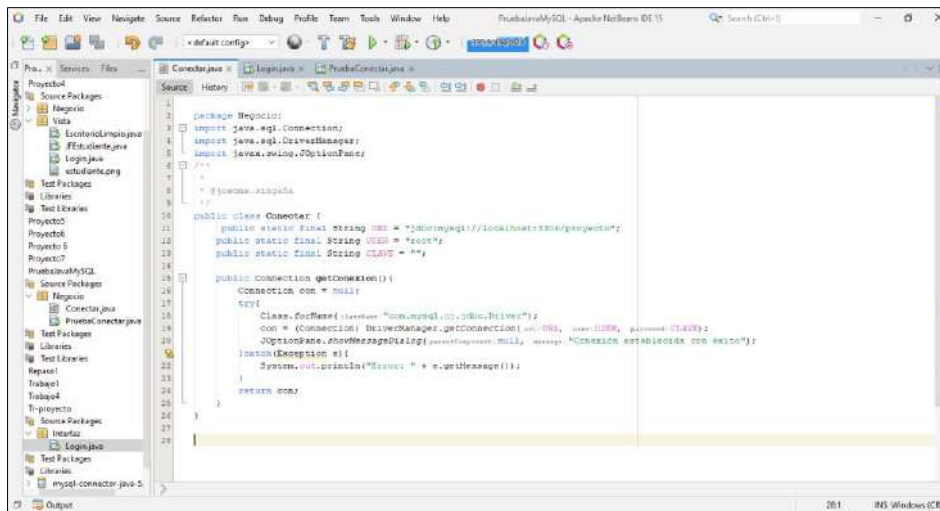
```

No olvidemos añadir los import al inicio de nuestra clase.

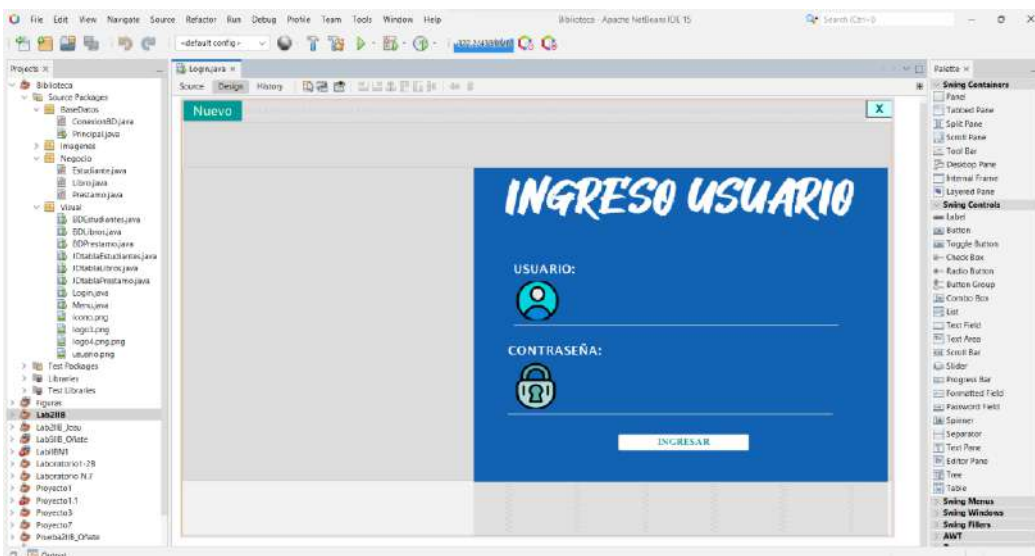
```

import java.sql.Connection;
import javax.swing.JOptionPane;

```



Se agrega JFrame llamado login que tendrá un JButton de conectar y un JLabel estado. Además de dos text Field para ingresar usuario y contraseña.



En el código del JFrame se agrega:

```

import BaseDatos.ConexionBD;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Toolkit;
import java.sql.Connection;
import java.util.Arrays;
import javax.swing.ImageIcon;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

```

Declaramos los datos de conexión a la base de datos.

```

Class.forName("com.mysql.jdbc.Driver");
cn=
DriverManager.getConnection("jdbc:mysql://localhost:3306/base_libros","root","root");

```

Función que va conectarse a la base de datos de mysql

```

java.sql.Connection cn;

```

Nos conectamos a la base de datos.

```

con= (Connection) DriverManager.getConnection(url, user, pass);

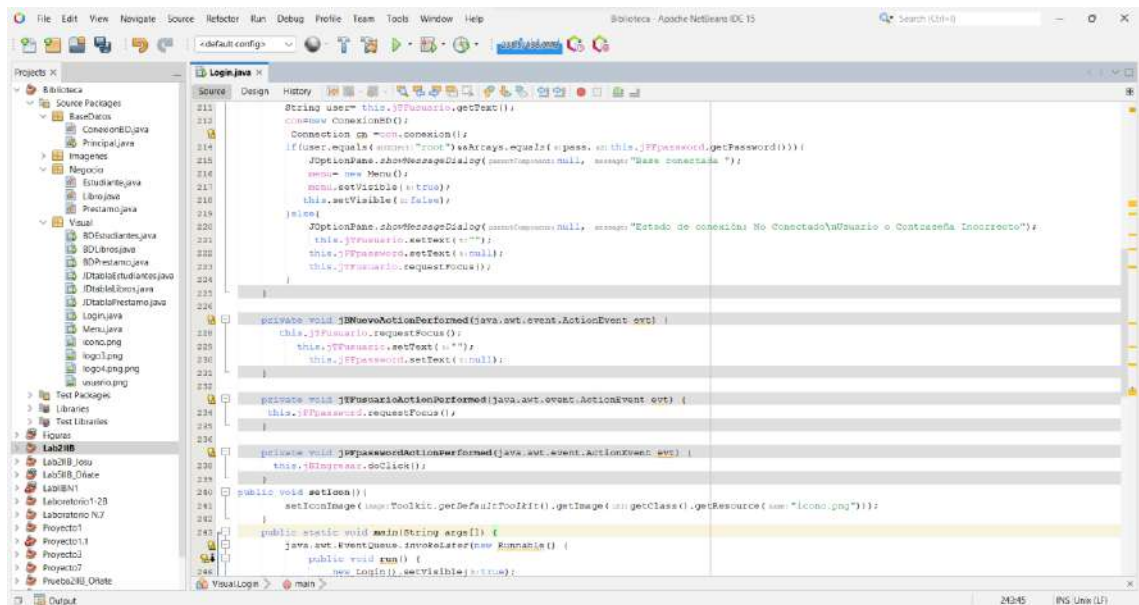
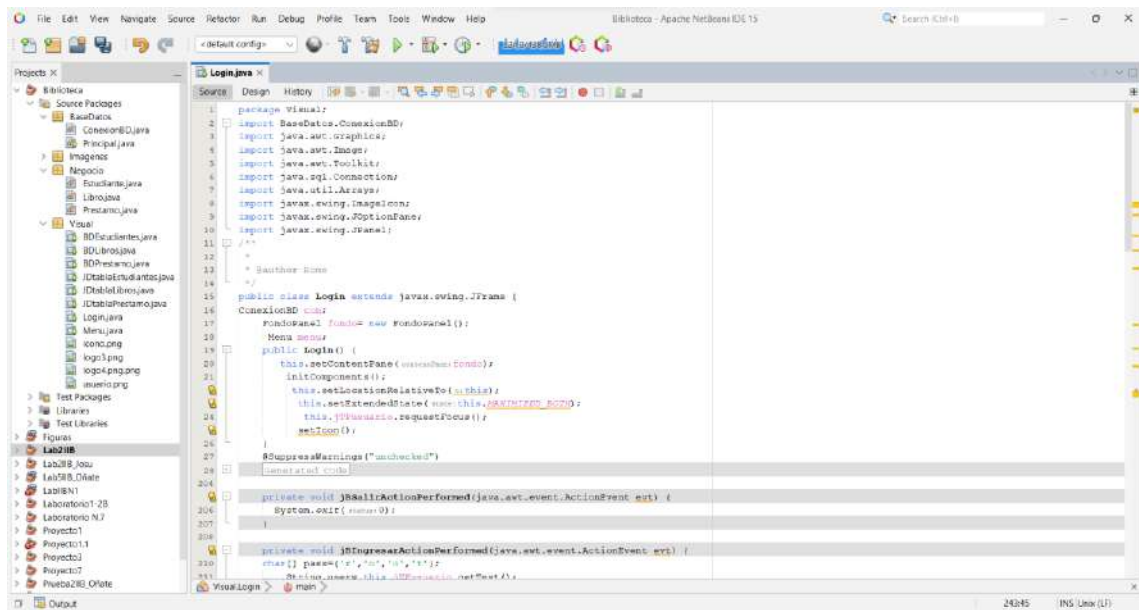
```

Si la conexión fue exitosa mostramos un mensaje de base conectada

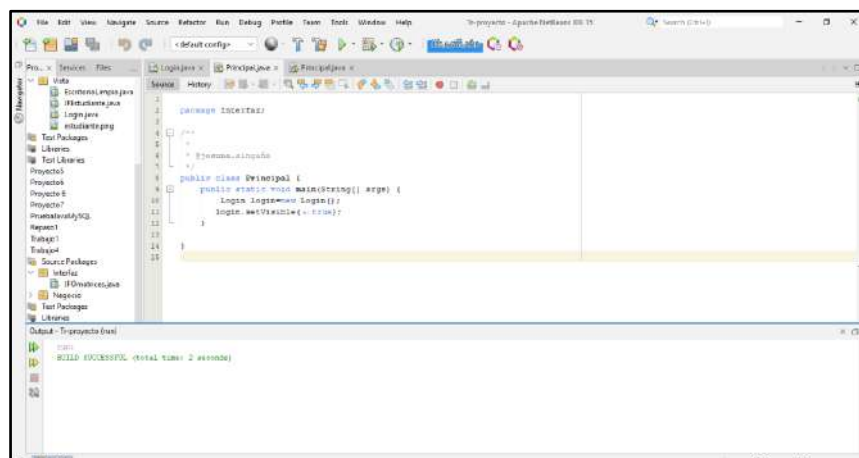
```

public Connection conexion() {
try{
Class.forName("com.mysql.jdbc.Driver");
cn=
DriverManager.getConnection("jdbc:mysql://localhost:3306/base_libros","root","root");
System.out.println("Se hizo la conexión de forma correcta");
}catch(Exception e){
System.out.println(e.getMessage());
}return cn;
}
Statement createStatement(){
throw new UnsupportedOperationException("No soportado");
}}

```

Configuración clase principal:



PAQUETES Y CLASES EN EL PROYECTO BIBLIOTECA

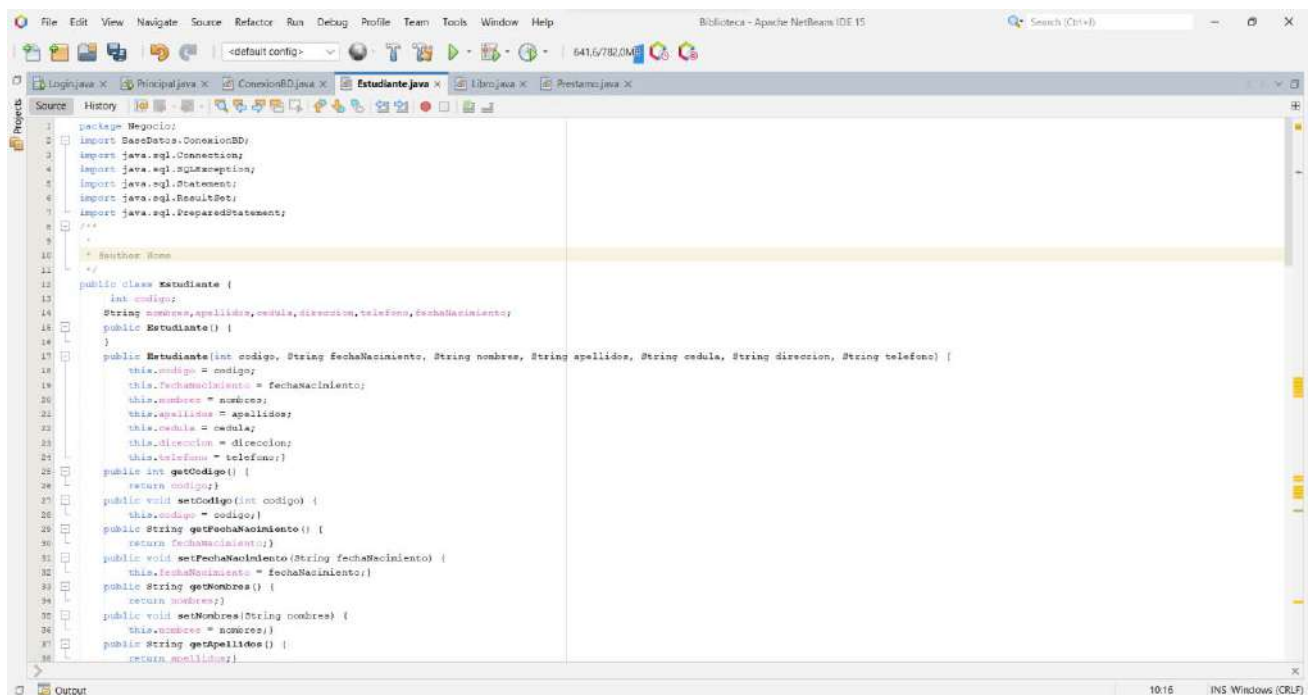
Crear las entidades (y cualquier otra entidad que tengan en su diseño): con botones para nuevo, actualizar borrar. cada ventana debe en la parte inferior incluir un objeto table que permita ver los cambios en la tabla correspondiente.

Paquete Negocio

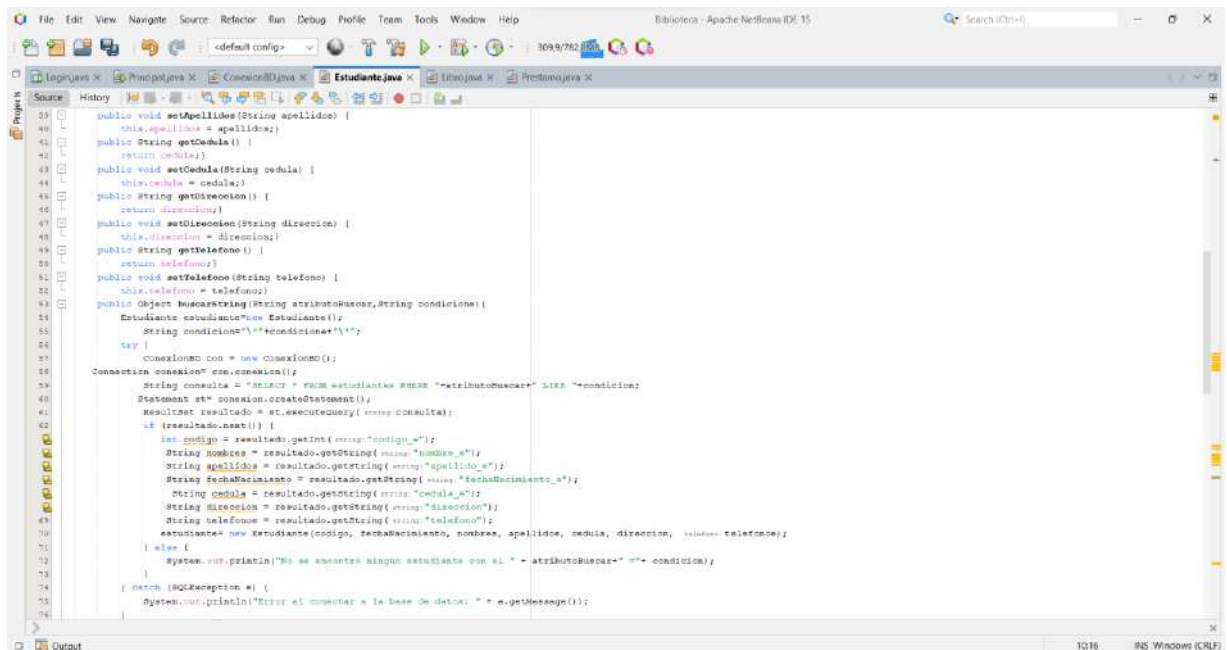
Lo integran las entidades de los registros de tablas como estudiante, libro y préstamo

Estudiante:

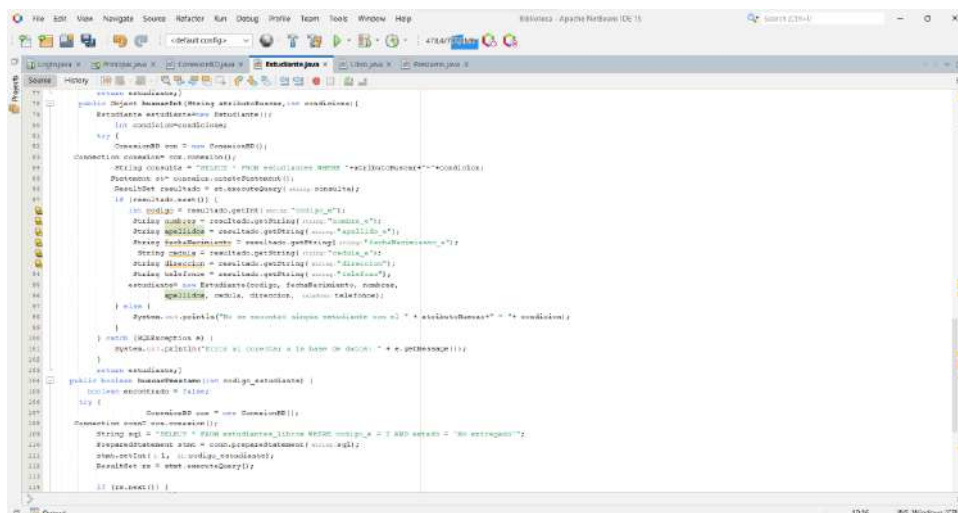
En la clase estudiante utilizamos varios import que nos van a ayudar con la conexión hacia MySQL, iniciamos declarando la clase y las variables que vamos a utilizar en la misma, acto seguido introducimos los constructores y también los getters y setters de cada uno de los atributos como código del estudiante, fecha de nacimiento, nombres, apellidos, cedula, dirección y teléfono para poder tomar los datos desde la clase para la parte visual, agregamos un objeto de buscarString conectado directamente con mysql para que pueda tomar los atributos de estudiantes y compararlos con la tabla y poder hacer una búsqueda efectiva hacia las tablas ya creadas, además creamos un método que nos ayude a comprobar si el estudiante se encuentra en la tabla de préstamos de tal manera que no podamos eliminarlo si consta en la misma.



```
1 package Negocio;
2 import BaseDatos.ConexionBD;
3 import java.sql.Connection;
4 import java.sql.DriverManager;
5 import java.sql.SQLException;
6 import java.sql.Statement;
7 import java.sql.ResultSet;
8 import java.sql.PreparedStatement;
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
```



```
39 public void setApellido(String apellido) {
40     this.apellido = apellido;
41 }
42 public String getApellido() {
43     return apellido;
44 }
45 public void setCedula(String cedula) {
46     this.cedula = cedula;
47 }
48 public String getDireccion() {
49     return direccion;
50 }
51 public void setDireccion(String direccion) {
52     this.direccion = direccion;
53 }
54 public String getTelefono() {
55     return telefono;
56 }
57 public void setTelefono(String telefono) {
58     this.telefono = telefono;
59 }
60 public Object buscarEstudiante(String atributoBuscar, String condicion) {
61     Estudiante estudiante = null;
62     String condicionSQL = "" + atributoBuscar + " " + condicion;
63     try {
64         ConexionBD con = new ConexionBD();
65         Connection conexion = con.conexion();
66         String consulta = "SELECT * FROM estudiantes WHERE " + atributoBuscar + " LIKE " + condicion;
67         Statement st = conexion.createStatement();
68         ResultSet resultado = st.executeQuery(consulta);
69         if (resultado.next()) {
70             int codigo = resultado.getInt("codigo");
71             String nombre = resultado.getString("nombre");
72             String apellido = resultado.getString("apellido");
73             String fechaNacimiento = resultado.getString("fechaNacimiento");
74             String cedula = resultado.getString("cedula");
75             String direccion = resultado.getString("direccion");
76             String telefono = resultado.getString("telefono");
77             estudiante = new Estudiante(codigo, fechaNacimiento, nombre, apellido, cedula, direccion, telefono);
78         } else {
79             System.out.println("No se encontró ningún estudiante con el " + atributoBuscar + " " + condicion);
80         }
81     } catch (SQLException e) {
82         System.out.println("Error al conectar a la base de datos: " + e.getMessage());
83     }
84 }
85 }
```



```
17 public void setTitulo(String titulo) {
18     this.titulo = titulo;
19 }
20 public String getTitulo() {
21     return titulo;
22 }
23 public void setAutor(String autor) {
24     this.autor = autor;
25 }
26 public String getAutor() {
27     return autor;
28 }
29 public void setCodigo(int codigo) {
30     this.codigo = codigo;
31 }
32 public int getCodigo() {
33     return codigo;
34 }
35 public void setFechaPublicacion(String fechaPublicacion) {
36     this.fechaPublicacion = fechaPublicacion;
37 }
38 public String getFechaPublicacion() {
39     return fechaPublicacion;
40 }
41 public void setCategorias(String categorias) {
42     this.categorias = categorias;
43 }
44 public String getCategorias() {
45     return categorias;
46 }
47 public void setReservado(boolean reservado) {
48     this.reservado = reservado;
49 }
50 public boolean getReservado() {
51     return reservado;
52 }
53 public Object buscarLibro(String atributoBuscar, String condicion) {
54     Libro libro = null;
55     String condicionSQL = "" + atributoBuscar + " " + condicion;
56     try {
57         ConexionBD con = new ConexionBD();
58         Connection conexion = con.conexion();
59         String consulta = "SELECT * FROM libros WHERE " + atributoBuscar + " LIKE " + condicion;
60         Statement st = conexion.createStatement();
61         ResultSet resultado = st.executeQuery(consulta);
62         if (resultado.next()) {
63             int codigo = resultado.getInt("codigo");
64             String titulo = resultado.getString("titulo");
65             String autor = resultado.getString("autor");
66             String fechaPublicacion = resultado.getString("fechaPublicacion");
67             String categorias = resultado.getString("categorias");
68             boolean reservado = resultado.getBoolean("reservado");
69             libro = new Libro(codigo, titulo, autor, fechaPublicacion, categorias, reservado);
70         } else {
71             System.out.println("No se encontró ningún libro con el " + atributoBuscar + " " + condicion);
72         }
73     } catch (SQLException e) {
74         System.out.println("Error al conectar a la base de datos: " + e.getMessage());
75     }
76 }
77 public void actualizarDatos(int codigo, String nuevoTitulo, String nuevoAutor, String nuevaFechaPublicacion, String nuevasCategorias, boolean nuevoReservado) {
78     try {
79         ConexionBD con = new ConexionBD();
80         Connection conexion = con.conexion();
81         String sql = "UPDATE libros SET titulo = '" + nuevoTitulo + "', autor = '" + nuevoAutor + "', fechaPublicacion = '" + nuevaFechaPublicacion + "', categorias = '" + nuevasCategorias + "', reservado = " + (nuevoReservado ? "true" : "false") + " WHERE codigo = " + codigo;
82         PreparedStatement stmt = con.prepareStatement(sql);
83         stmt.executeUpdate();
84     } catch (SQLException e) {
85         System.out.println("Error al actualizar los datos del libro: " + e.getMessage());
86     }
87 }
```

Libro:

En la clase libro de igual manera que en estudiante utilizamos los import para poder lograr la conexión con MYSQL y empezamos declarando la clase y declarando los atributos que se van a utilizar en la misma como código, titulo, autor, el número disponible de copias y el número total de copias, agregamos un constructor y los getters y setters para cada uno de los atributos ya mencionados, seguido de eso procedemos a ingresar los métodos para hacer la conexión con MYSQL y poder comparar los datos con las tablas ya creadas de tal forma de buscar el libro por cualquiera de sus atributos tanto como int y como string. También agregamos un método actualizarDatos el cuál con la función UPDATE nos ayudará a actualizar los datos de la tabla que así requiera el usuario, por último agregamos un método para comparar si el libro consta de la tabla prestamos al momento de requerir eliminarlo para poder hacer una eliminación adecuada y correcta.

```
import java.sql.*;
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class Libro {
    private int idLibro;
    private String titulo;
    private String autor;
    private int copias;
    private int prestados;

    public Libro() {}

    public Libro(int idLibro, String titulo, String autor, int copias, int prestados) {
        this.idLibro = idLibro;
        this.titulo = titulo;
        this.autor = autor;
        this.copias = copias;
        this.prestados = prestados;
    }

    public int getIdLibro() {
        return idLibro;
    }

    public void setIdLibro(int idLibro) {
        this.idLibro = idLibro;
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    public int getCopias() {
        return copias;
    }

    public void setCopias(int copias) {
        this.copias = copias;
    }

    public int getPrestados() {
        return prestados;
    }

    public void setPrestados(int prestados) {
        this.prestados = prestados;
    }
}
```

```
public void conectar() {
    try {
        Class.forName("com.mysql.jdbc.Driver");
        Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/biblioteca", "root", "");
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery("SELECT * FROM libro WHERE titulo = '" + titulo + "'");
        while (rs.next()) {
            idLibro = rs.getInt("idLibro");
            titulo = rs.getString("titulo");
            autor = rs.getString("autor");
            copias = rs.getInt("copias");
            prestados = rs.getInt("prestados");
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }
}

public void actualizarLibro(int idLibro, String titulo, String autor, int copias, int prestados) {
    try {
        Connection con = new Connection();
        con.conectar();
        String sql = "UPDATE libro SET titulo = '" + titulo + "', autor = '" + autor + "', copias = '" + copias + "', prestados = '" + prestados + "' WHERE idLibro = " + idLibro;
        Statement st = con.createStatement();
        st.executeUpdate(sql);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }
}
```

```
public void eliminarLibro(int idLibro) {
    try {
        Connection con = new Connection();
        con.conectar();
        String sql = "DELETE FROM libro WHERE idLibro = " + idLibro;
        Statement st = con.createStatement();
        st.executeUpdate(sql);
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }
}

public void buscarLibro(String titulo) {
    try {
        Connection con = new Connection();
        con.conectar();
        String sql = "SELECT * FROM libro WHERE titulo = '" + titulo + "'";
        Statement st = con.createStatement();
        ResultSet rs = st.executeQuery(sql);
        while (rs.next()) {
            idLibro = rs.getInt("idLibro");
            titulo = rs.getString("titulo");
            autor = rs.getString("autor");
            copias = rs.getInt("copias");
            prestados = rs.getInt("prestados");
        }
    } catch (SQLException e) {
        JOptionPane.showMessageDialog(null, e.getMessage());
    }
}
```

Préstamo:

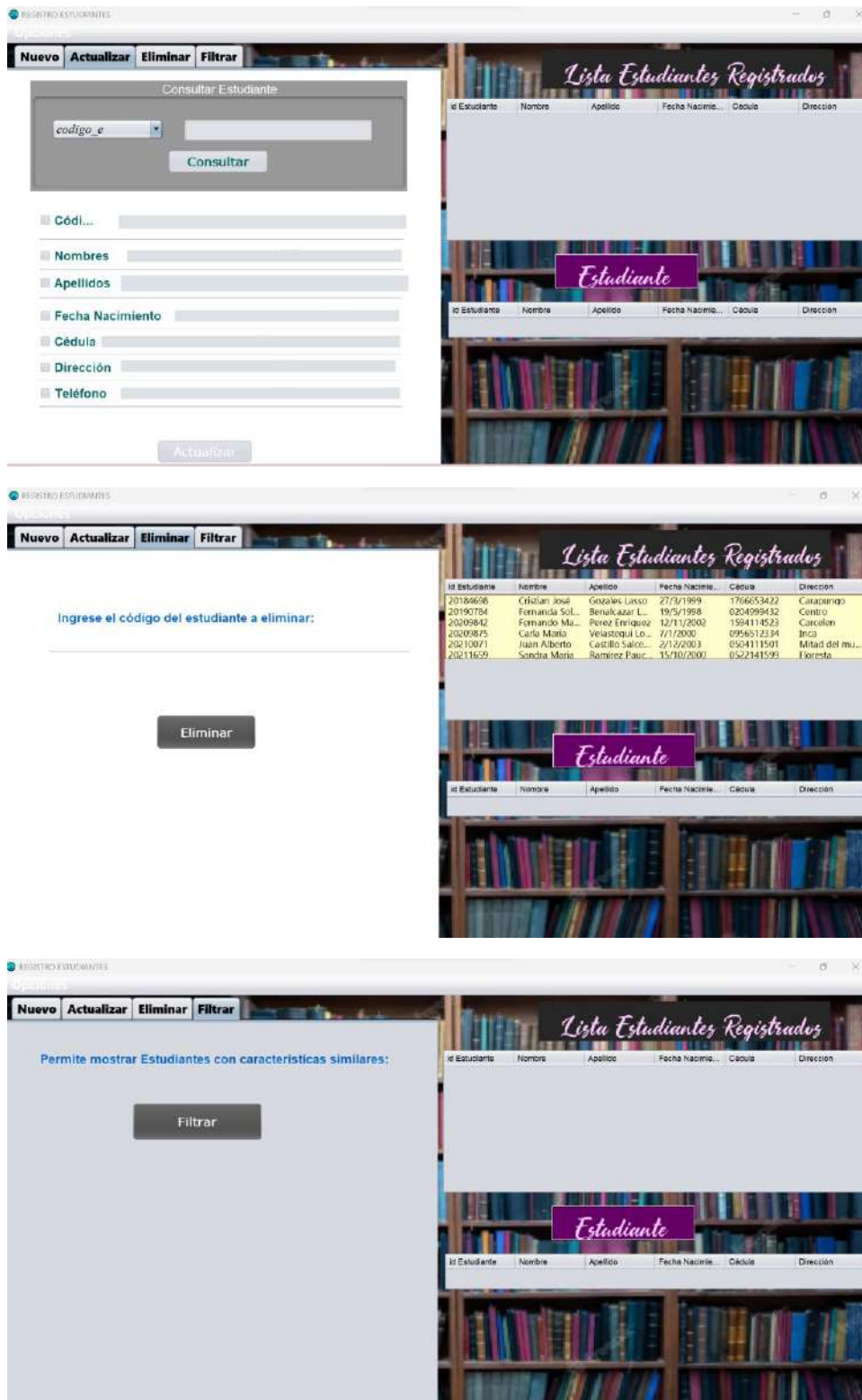
Por último en la parte de préstamo de igual manera empezamos con los import de la BaseDatos y de los demás necesarios para conectar con MYSQL, declaramos la clase y sus métodos correspondientes en este caso serían código del estudiante, código del libro, fecha del préstamo, el estado del préstamo y la fecha de entrega, añadimos un constructor y agregamos los getters y setters para cada uno de los atributos ya antes mencionados en la

[illegible]

```

1  import java.sql.*;
2
3  public class Piscina {
4
5      // Método para conectar a la base de datos
6      private static Connection getConnection() throws SQLException {
7          String url = "jdbc:mysql://localhost:3306/piscina";
8          String user = "root";
9          String pass = "root";
1         try {
12             Connection conn = DriverManager.getConnection(url, user, pass);
13             return conn;
14         } catch (SQLException e) {
15             throw new RuntimeException("Error al conectar a la base de datos: " + e.getMessage());
16         }
17     }
18
19     // Método para buscar un libro por título
20     private static void buscarLibro(String titulo) throws SQLException {
21         Connection conn = getConnection();
22         String sql = "SELECT * FROM libros WHERE titulo = ?";
23         PreparedStatement stmt = conn.prepareStatement(sql);
24         stmt.setString(1, titulo);
25         ResultSet rs = stmt.executeQuery();
26         while (rs.next()) {
27             System.out.println("Se encontró el libro: " + rs.getString("titulo"));
28             System.out.println("Autor: " + rs.getString("autor"));
29             System.out.println("Categoría: " + rs.getString("categoria"));
30             System.out.println("Precio: " + rs.getDouble("precio"));
31             System.out.println("Disponibilidad: " + rs.getBoolean("disponible"));
32         }
33     }
34
35     // Método principal
36     public static void main(String[] args) {
37         if (args.length < 1) {
38             System.out.println("Uso: java Piscina <título>");
39             return;
40         }
41         String titulo = args[0];
42         try {
43             buscarLibro(titulo);
44         } catch (SQLException e) {
45             e.printStackTrace();
46         }
47     }
48 }

```

En este JFrame utilizamos tres jTable y un tabbed pane que nos permite crear las pestañas nuevo, actualizar, filtrar y eliminar por las que el usuario es capaz de navegar, en la pestaña nuevo tenemos siete labels para los títulos código estudiante, nombres, apellido, fecha nacimiento, cédula, dirección, teléfono, cada uno con su respectivo text field, utilizados para que el usuario sea capaz de ingresar datos, además tenemos un button con el título agregar.

Código:

```

private void jBAgregarActionPerformed(java.awt.event.ActionEvent evt) {
    Estudiante estudiante= new Estudiante();
    try{
        estudiante=new Estudiante(Integer.parseInt(this.jTFcodigo.getText()),
this.jTFFecha.getText(),
        this.jTFnombre.getText(), this.jTFApellido.getText(), this.jTFcedula.getText(),
        this.jTFdireccion.getText(), this.jTFtelefono.getText());
        PreparedStatement pps= cn.prepareStatement("INSERT INTO estudiantes
VALUES(?,?,?,?,?,?,?)");
        pps.setInt(1, estudiante.getCodigo());
        pps.setString(2, estudiante.getNombres());
        pps.setString(3, estudiante.getApellidos());
        pps.setString(4, estudiante.getFechaNacimiento());
        pps.setString(5, estudiante.getCedula());
        pps.setString(6, estudiante.getDireccion());
        pps.setString(7, estudiante.getTelefono());
        pps.executeUpdate();

        JOptionPane.showMessageDialog(null, "Datos guardados correctamente");
        this.jLEstudiante.setText("Agregado");
        jTEstudiante.setModel(tabla(estudiante));
    } catch (SQLException ex) {
        Logger.getLogger(BDEstudiantes.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(null, "Ocurrio un error al ingresar los datos");
    }
    eliminar(jTLista);
    listar();
}

```

Para la pestaña actualizar utilizamos un combobox con las opciones codigo_e y cedula_e, también utilizamos un text field, dos button con los títulos respectivos de consultar, actualizar, y siete check box con los títulos código, nombres, apellidos, fecha nacimiento, cédula, dirección, teléfono, cada uno con un text field a su lado derecho.

Código:

```
private void jBAActualizarActionPerformed(java.awt.event.ActionEvent evt) {  
    String consulta = "";  
    ArrayList<String> atributosActualizar = new ArrayList<String>();  
    String dato;  
    if (cheaCodigo.isSelected()||cheaNombres.isSelected()||cheaApellidos.isSelected()||  
cheafecha.isSelected()||cheaCedula.isSelected()  
        ||cheaDireccion.isSelected()|| cheaTelefono.isSelected()){  
        if (cheaCodigo.isSelected()){ atributosActualizar.add("codigo_e="+txtaCodigo.getText());}  
        if  
(cheaNombres.isSelected()){ atributosActualizar.add("nombre_e=\""+txtaNombres.getText()  
+"\"");}  
        if  
(cheaApellidos.isSelected()){ atributosActualizar.add("apellido_e=\""+txtaApellidos.getText  
()+"\"");}  
        if  
(cheafecha.isSelected()){ atributosActualizar.add("fechaNacimiento_e=\""+txtaFecha.getTex  
t()+"\"");}  
        if  
(cheaCedula.isSelected()){ atributosActualizar.add("cedula_e=\""+txtaCedula.getText()+"\"  
");}  
        if  
(cheaDireccion.isSelected()){ atributosActualizar.add("direccion=\""+txtaDireccion.getText(  
)+"\"");}  
        if  
(cheaTelefono.isSelected()){ atributosActualizar.add("telefono=\""+txtaTelefono.getText()+"  
\"");}  
        Iterator i = atributosActualizar.iterator();  
        while(i.hasNext()){  
            consulta+= i.next()+",";  
        }  
        consulta= consulta.substring(0,consulta.length()-1);  
        String sql;  
        if(this.jCBconsultar.equals("codigo_e")){  
            dato=this.jTFBuscar.getText();  
        }else{
```

```

        dato="\""+this.jTFBuscar.getText()+"\"";
    }

    sql="UPDATE estudiantes SET " + consulta+" WHERE
"+this.jCBconsultar.getSelectedItem()+"="+dato;

    try{
        PreparedStatement pps= cn.prepareStatement(sql);
        pps.executeUpdate();

        JOptionPane.showMessageDialog(null, "Datos Actualizados correctamente");

        Estudiante estudiante = new Estudiante();

        estudiante = (Estudiante)
        estudiante.buscarInt("codigo_e",Integer.parseInt(this.txtaCodigo.getText()));

        this.jPAActual.setVisible(true);

        this.jTAActual.setModel(tablaA(estudiante));

    }catch(SQLException ex){

        Logger.getLogger(BDEstudiantes.class.getName()).log(Level.SEVERE,null,ex);

        JOptionPane.showMessageDialog(null, ex);

        JOptionPane.showMessageDialog(null,"Ocurrio un error al intentar Actualizar.");

    }

    eliminar(jTLista);

    listar();

    }else{

        JOptionPane.showMessageDialog(null, "No se puede actualizar porque no hubo
        cambio");}

    }

```

En la pestaña eliminar tenemos un label con el texto Ingrese el código del estudiante a eliminar, además hay un text field y un button con el título eliminar.

```

private void jBEliminarActionPerformed(java.awt.event.ActionEvent evt) {

    Estudiante estudiante = new Estudiante();

    estudiante = (Estudiante)
    estudiante.buscarInt("codigo_e",Integer.parseInt(this.txteCodigo.getText()));

    try{

```

```

        if (estudiante == null ||
estudiante.buscarPrestamo(Integer.parseInt(this.txteCodigo.getText()))==true) {
            if (estudiante == null){
                JOptionPane.showMessageDialog(null, "No se ha encontrado al estudiante");
                this.txteCodigo.setText("");
                this.txteCodigo.requestFocus();
            }else{
                JOptionPane.showMessageDialog(null, "El estudiante no puede ser eliminado
tiene préstamos pendientes");
            }

        } else {
            this.txteCodigo.setEnabled(false);
            this.jLEstudiante.setText("ENCONTRADO");
            this.jTEstudiante.setModel(tabla(estudiante));

            int opcion;

            opcion=JOptionPane.showConfirmDialog(null, "Desea ELIMINAR el estudiante",
"Confirmación usuario",
            JOptionPane.YES_NO_OPTION);
            if(opcion==JOptionPane.YES_OPTION){
                PreparedStatement pps=cn.prepareStatement("DELETE FROM estudiantes
WHERE codigo_e = "+txtaCodigo.getText());
                pps.executeUpdate();
                JOptionPane.showMessageDialog(null, "Registro Eliminado
Correctamente.", "Eliminar",JOptionPane.INFORMATION_MESSAGE);
                this.jLEstudiante.setText("ELIMINADO");
            }
        }
        eliminar(jTLista);
        listar();
    } catch (SQLException ex){
        Logger.getLogger(BDEstudiantes.class.getName()).log(Level.SEVERE,null,ex);
    }
}

```

```

JOptionPane.showMessageDialog(null, "Ocurrio un error al intentar Eliminar");
    }
}

```

En la pestaña filtrar tenemos un label con el texto permite mostrar estudiantes con características similares, y un button con el título filtrar.

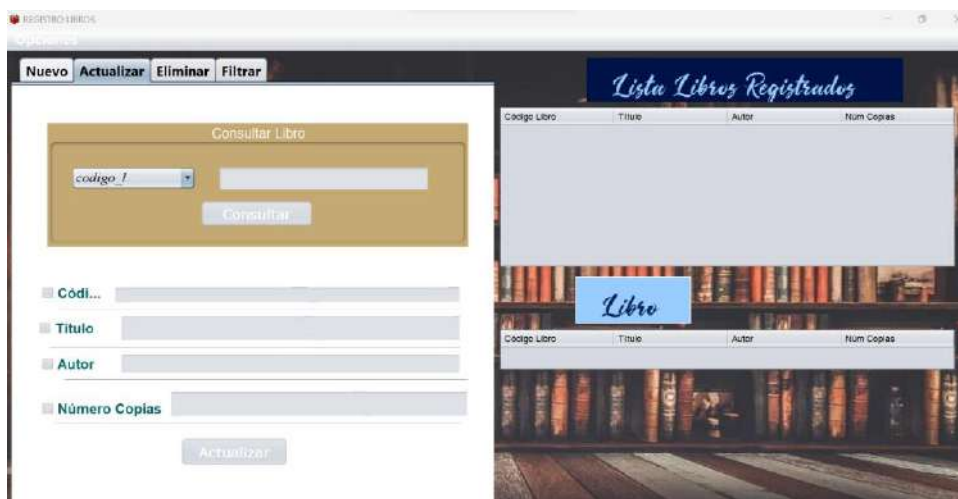
Código:

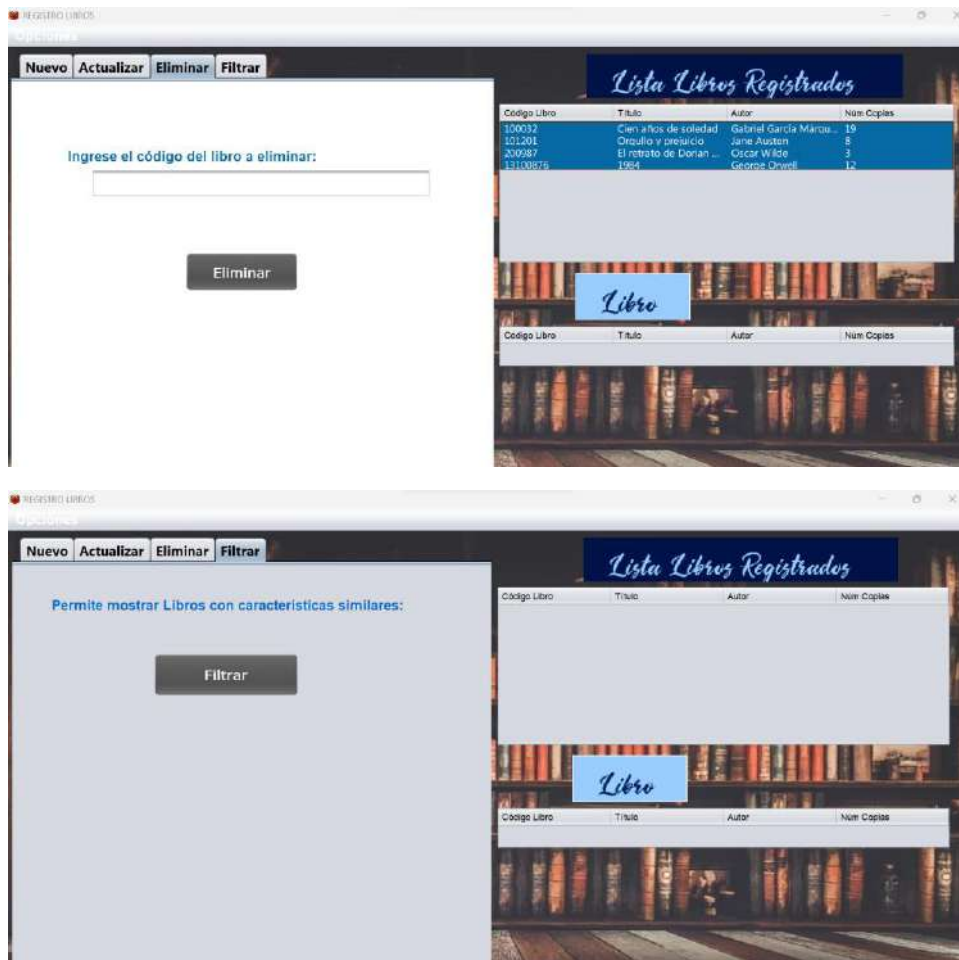
```

private void jBFiltrarActionPerformed(java.awt.event.ActionEvent evt) {
    JTablaEstudiantes jdtablaestudiante=new JTablaEstudiantes();
    jdtablaestudiante.setVisible(true);
}

```

BDLibros:





En este JFrame utilizamos tres jTable y un tabbed pane que nos permite crear las pestañas nuevo, actualizar, filtrar y eliminar por las que el usuario es capaz de navegar, en la pestaña nuevo tenemos cuatro labels para los títulos código libro, título, autor, número de copias cada uno con su respectivo text field, utilizados para que el usuario sea capaz de ingresar datos, además tenemos un button con el título agregar.

Código:

```
private void jBAgregarActionPerformed(java.awt.event.ActionEvent evt) {  
    Libro libro=new Libro();  
    try{  
        libro=new Libro(Integer.parseInt(this.jTFcodigoL.getText()),  
            this.jTFtitulo.getText(),  
            this.jTFAutor.getText(),Integer.parseInt(this.jTFcopias.getText()),Integer.parseInt(this.jTFCopias.getText()));  
        PreparedStatement pps= cn.prepareStatement("INSERT INTO libros  
VALUES(?,?,?,?)");  
        pps.setInt(1, libro.getCodigo());
```

```

pps.setString(2, libro.getTitulo());
pps.setString(3, libro.getAutor());
pps.setInt(4, libro.getCopias());
pps.setInt(5, libro.getTotal());
pps.executeUpdate();

JOptionPane.showMessageDialog(null, "Datos guardados correctamente");
this.jLlibro.setText("Agregado");
jTLibro.setModel(tablaLibro(libro));
} catch (SQLException ex) {
    Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE, null, ex);
    JOptionPane.showMessageDialog(null, "Ocurrio un error al ingresar los datos");
}
eliminarTablaLibros(jTListaLibros);
listarLibros();

}

```

En la pestaña actualizar tenemos un combo box con las opciones odigo_1, titulo_1, autor_1, un text field, dos button con el título consultar, actualizar respectivamente, y cuatro check box con los títulos código, título, autor, número de copias, cada uno con su respectivo text field.

Código:

```

private void jBConsultarActionPerformed(java.awt.event.ActionEvent evt) {
    Libro libro=new Libro();
    if(jCBconsultar.getSelectedItem()=="codigo_1"||jCBconsultar.getSelectedItem()=="copias_1")
    {
        libro = (Libro)
        libro.buscarIntLibro(jCBconsultar.getSelectedItem().toString(),Integer.parseInt(this.jTFBusc
        ar.getText()));
        if (libro == null) {
            JOptionPane.showMessageDialog(null, "No se ha encontrado el libro");
            this.jTFBuscar.setText("");
            this.jTFBuscar.requestFocus();

```

```

    } else {
        if(libro.getCodigo()==0){
            JOptionPane.showMessageDialog(null, "No se ha encontrado el libro con
"+jCBconsultar.getSelectedItem()+" = "+jTFBuscar.getText());
            this.jTFBuscar.setText("");
            this.jTFBuscar.requestFocus();
        }else{
            this.jTFBuscar.setEnabled(false);
            infoActualizarLibro(libro);
            this.jLlibro.setText("ENCONTRADO");
            jTLibro.setModel(tablaLibro(libro));
            this.jBActualizar.setEnabled(true);

            this.txtaCodigol.setEnabled(false);
            habilitarLibro();
        }
    }
}
}else{
    libro = (Libro)
libro.buscarStringLibro(jCBconsultar.getSelectedItem().toString(),this.jTFBuscar.getText())
;

    if (libro == null) {
        JOptionPane.showMessageDialog(null, "No se ha encontrado el libro");
        this.jTFBuscar.setText("");
        this.jTFBuscar.requestFocus();
    } else {
        this.jTFBuscar.setEnabled(false);
        infoActualizarLibro(libro);
        this.jLlibro.setText("ENCONTRADO");
        jTLibro.setModel(tablaLibro(libro));
        this.jBActualizar.setEnabled(true);
    }
}

```



```

        this.txtaCodigol.setEnabled(false);
        habilitarLibro();
    }
}

}

private void jBActualizarActionPerformed(java.awt.event.ActionEvent evt) {
String dato;

    ArrayList<String> atributosActualizar = new ArrayList<String>();
String consulta = "";
if (cheaCodigol.isSelected()||cheaTitulo.isSelected()||cheaAutor.isSelected()||
    cheaCopias.isSelected()){
if (cheaCodigol.isSelected()){ atributosActualizar.add("codigo_l="+txtaCodigol.getText());}
if
(cheaTitulo.isSelected()){ atributosActualizar.add("titulo_l='"+txtaTitulo.getText()+"'");}
if (cheaAutor.isSelected()){ atributosActualizar.add("autor_l=
='"+txtaAutor.getText()+"'");}
if (cheaCopias.isSelected()){ atributosActualizar.add("copias_l="+txtaCopias.getText());
atributosActualizar.add("total_l="+txtaCopias.getText());}

Iterator i = atributosActualizar.iterator();

while(i.hasNext()){
    consulta+= i.next()+",";

}

consulta= consulta.substring(0,consulta.length()-1);

String sql;
if(this.jCBconsultar.equals("codigo_l")){
    dato=this.jTFBuscar.getText();
}else{

```

```

        dato="\""+this.jTFBuscar.getText()+"\"";
    }

    sql="UPDATE libros SET " + consulta+" WHERE
"+this.jCBconsultar.getSelectedItem()+"="+dato;

    try{
        PreparedStatement pps= cn.prepareStatement(sql);
        pps.executeUpdate();
        JOptionPane.showMessageDialog(null, "Datos Actualizados correctamente");

        Libro libro = new Libro();
        libro =(Libro) libro.buscarIntLibro("codigo_l",Integer.parseInt(this.txtaCodigol.getText()));
        this.jPActual.setVisible(true);
        this.jTActual.setModel(tablaLibroA(libro));

    }catch(SQLException ex){
        Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);
        JOptionPane.showMessageDialog(null, ex);
        JOptionPane.showMessageDialog(null,"Ocurrio un error al intentar Actualizar.");
    }
    eliminarTablaLibros(jTListaLibros);
    listarLibros();
}else{
    JOptionPane.showMessageDialog(null, "No se puede actualizar porque no hubo
cambio");
}

}

```

En la pestaña eliminar tenemos un label con el texto ingrese el código del libro a eliminar, además hay un text field y un button con el título eliminar.

Código:

```

private void jBEliminarActionPerformed(java.awt.event.ActionEvent evt) {

```

```

Libro libro=new Libro();

        libro=(Libro) libro.buscarIntLibro("codigo_l",
Integer.parseInt(this.txteCodigo.getText()));

        try{

            if (libro == null||
libro.buscarPrestamo(Integer.parseInt(this.txteCodigo.getText()))==true) {

                if (libro == null){

                    JOptionPane.showMessageDialog(null, "No se encontrado el libro");

                    this.txteCodigo.setText("");

                    this.txteCodigo.requestFocus();

                }else{

                    JOptionPane.showMessageDialog(null, "El libro no puede ser eliminado está
presente en préstamos pendientes");

                }

            } else {

                this.txteCodigo.setEnabled(false);

                infoActualizarLibro(libro);

                this.jLlibro.setText("ENCONTRADO");

                this.jTLibro.setModel(tablaLibro(libro));

            }

            this.txtaCodigol.setEnabled(false);

            int opcion;

            opcion=JOptionPane.showConfirmDialog(null, "Desea ELIMINAR el libro",
"Confirmación usuario",
JOptionPane.YES_NO_OPTION);

            if(opcion==JOptionPane.YES_OPTION){

                PreparedStatement pps=cn.prepareStatement("DELETE FROM libros WHERE
codigo_l = "+txtaCodigol.getText());

                pps.executeUpdate();

                JOptionPane.showMessageDialog(null, "Registro Eliminado
Correctamente.", "Eliminar",JOptionPane.INFORMATION_MESSAGE);

                this.jLlibro.setText("ELIMINADO");

```

```

    }
}
eliminarTablaLibros(jTListaLibros);
listarLibros();
    } catch (SQLException ex){
        Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);
        JOptionPane.showMessageDialog(null, "Ocurrio un error al intentar Eliminar");
    }
}

```

En la pestaña filtrar hay un label con el texto Permite mostrar libros con características similares, junto a un button con el título filtrar.

Código:

```

private void jBFiltrarActionPerformed(java.awt.event.ActionEvent evt) {
    JDTablaLibros jdtablalibros=new JDTablaLibros();
    jdtablalibros.setVisible(true);
}

```

BDPréstamo:

Préstamo

Consultar

Entregar

Filtrar

Estudiante

Para acceder a un libro debe presentar un código de estudiante:

Código Estudiante:

Ingresar

Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Prestamo

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Préstamo

Consultar

Entregar

Filtrar

Consultar Préstamo

código /

Consultar

Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Prestamo

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Préstamo

Consultar

Entregar

Filtrar

Datos entrega

Código Estudiante:

Código Libro:

Fecha Entrega:

Entregar

Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
20210071	100012	27/2/2021	No Entregado	sin fecha

Prestamo

Código Estudiante	Código Libro	Fecha Prestamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------



En este frame tenemos tres table y un tabbed pane con las opciones préstamo, consultar, entregar, filtrar.

Cada uno de los table tiene características únicas, las cuales son detalladas con el código de cada uno.

```
public class BDPrestamo extends javax.swing.JFrame {
    ConexionBD con = new ConexionBD();
    Connection cn= con.conexion();
    BDPrestamo.FondoPanel fondo1= new BDPrestamo.FondoPanel();
    int contador=0;
    public BDPrestamo() {
        this.setContentPane(fondo1);
        initComponents();
        this.setLocationRelativeTo(null);
        this.setExtendedState(this.MAXIMIZED_BOTH);
        this.jBDisponibilidad.setVisible(true);
        this.jTFBuscar.setEnabled(false);
        this.jPPrestamoLibro.setVisible(false);
        this.jBDisponibilidad.setVisible(false);
        this.jPDisponibilidad.setVisible(false);
        setIcon();
    }
    this.jPPrestamoLibro.setVisible(false);
    this.jTFBuscar.setEnabled(false);
    this.jPDisponibilidad.setVisible(false);
```



```

this.jLPrestamo.setText("");
this.jPDisponibilidad.setVisible(false);
this.jTFBuscar.setText("");
this.jBDisponibilidad.setVisible(false);
eliminarTablaPrestamo(this.jTLibro);
eliminarTablaPrestamo(this.jTPrestamo);
eliminarTablaPrestamo(this.jTListaPrestamo);
switch(jTPOpciones.getSelectedIndex()){
    case 0:
        this.jTFcodigoE.setText(null);
        this.jTFcodigoL.setText(null);
        this.jTFfechap.setText(null);
        break;
    case 1:
        this.jTFBuscar.setText(null);
        break;
    case 2:
        this.jTFcodEntrega.setText(null);
        this.jTFcodigoLe.setText(null);
        this.jTFfechaE.setText(null);
        listarPrestamos();
        break;
}

```

PRESTAMO

En este caso para acceder a un libro, se debe ingresar el código único del estudiante, el código del libro y la fecha del préstamo, para actualizar la base de datos.

```

Prestamo prestamo = new Prestamo();
prestamo = (Prestamo) prestamo.buscarPrestamo("codigo_e","codigo_l",
    Integer.parseInt(this.jTFcodigoE.getText()),
    Integer.parseInt(this.jTFcodigoL.getText()));

```

```

if (prestamo.getCodigo_e() == 0) {
    int dato;
    try{
        prestamo=new Prestamo( Integer.parseInt(this.jTFcodigoE.getText()),
                                Integer.parseInt(this.jTFcodigoL.getText()),
                                this.jTFfechap.getText(),"No entregado","sin fecha");
        PreparedStatement pps= cn.prepareStatement("INSERT INTO estudiantes_libros
VALUES(?,?,?,?,?)");
        pps.setInt(1, prestamo.getCodigo_e());
        pps.setInt(2, prestamo.getCodigo_l());
        pps.setString(3, prestamo.getFechaPrestamo());
        pps.setString(4, prestamo.getEstado());
        pps.setString(5, prestamo.getFechaEntrega());
        pps.executeUpdate();
        JOptionPane.showMessageDialog(null, "Datos guardados correctamente");
        this.jLPrestamo.setText("Agregado");
        jTPrestamo.setModel(tablaPrestamo(prestamo));
        Libro libro2 = new Libro();
        libro2 =(Libro)
libro2.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoL.getText()));
        int num=libro2.getCopias()-1;
        try{
            String sql = "UPDATE libros SET copias_l = "+num +" WHERE
codigo_l="+this.jTFcodigoL.getText();
            PreparedStatement ppsp= cn.prepareStatement(sql);
            ppsp.executeUpdate();
            Libro libro = new Libro();
            libro =(Libro)
libro.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoL.getText()));
            this.jTLibro.setModel(tablaLibro(libro));
            this.jPDisponibilidad.setVisible(true);
        }catch(SQLException ex){
            Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);

```

```

        JOptionPane.showMessageDialog(null, ex);

        JOptionPane.showMessageDialog(null, "Ocurrio un error al intentar
Actualizar libro.");
    }

    } catch(SQLException ex){
        Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);
        JOptionPane.showMessageDialog(null, ex);
        JOptionPane.showMessageDialog(null, "Ocurrio un error al intentar Crear.");
    }
} else {
    jLPrestamo.setText("Registrado");
    jTPrestamo.setModel(tablaPrestamo(prestamo));
    if(prestamo.getEstado().equalsIgnoreCase("No Entregado")){
        JOptionPane.showMessageDialog(null, "NO puede realizar el préstamo porque ya
existe");
    } else{
        int opcion;

        opcion=JOptionPane.showConfirmDialog(null, "Quiere volver a alquilar?",
"Confirmación usuario",
        JOptionPane.YES_NO_OPTION);

        if(opcion==JOptionPane.YES_OPTION){
            String sql ="UPDATE estudiantes_libros SET fecha_p
= \""+jTFfechap.getText()+"\", estado= \"No Entregado\", fecha_e= \"sin fecha\" WHERE
codigo_e= \""+this.jTFcodigoE.getText()+" AND codigo_l= \""+this.jTFcodigoL.getText();

            try{
                PreparedStatement ppsa= cn.prepareStatement(sql);
                ppsa.executeUpdate();

                Libro libro2 = new Libro();

                libro2 =(Libro)
libro2.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoL.getText()));

                int num=libro2.getCopias()-1;

                Libro libro = new Libro();

                libro =(Libro)
libro.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoL.getText()));

```

```

        this.jTLibro.setModel(tablaLibro(libro));
        this.jPDisponibilidad.setVisible(true);
        this.jLPrestamo.setText("Registrado");
        prestamo = (Prestamo) prestamo.buscarPrestamo("codigo_e","codigo_l",
        Integer.parseInt(this.jTFcodigoE.getText()),
        Integer.parseInt(this.jTFcodigoL.getText()));
    }catch(SQLException ex){
        Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);
        JOptionPane.showMessageDialog(null, ex);
        JOptionPane.showMessageDialog(null,"Ocurrio un error");
    }
}
jLPrestamo.setText("Registrado");
jTPrestamo.setModel(tablaPrestamo(prestamo));
}
}
eliminarTablaPrestamo(jTListaPrestamo);
listarPrestamos();

```

CONSULTAR

En consultar disponemos de un combo box un textfield que toma la informacion para consultar el codigo del libro o del estudiante y verificar si el estudiante tiene prestamos previos o la disponibilidad del libro.

```
Prestamo prestamo=new Prestamo();
```

```

if(jCBconsultar.getSelectedItem()=="codigo_e"||jCBconsultar.getSelectedItem()=="codigo_
l"){
    prestamo = (Prestamo)
    prestamo.buscarIntPrestamo(jCBconsultar.getSelectedItem().toString(),Integer.parseInt(this.
    jTFBuscar.getText()));
    if (prestamo == null) {
        JOptionPane.showMessageDialog(null, "No se ha encontrado el préstamo");
        this.jTFBuscar.setText(null);
        this.jTFBuscar.requestFocus();
    }
}

```

```

    } else {
        if(prestamo.getCodigo_l()==0){
            JOptionPane.showMessageDialog(null, "No se ha encontrado el préstamo con
            "+jCBconsultar.getSelectedItemAt()+" = "+jTFBuscar.getText());
            this.jTFBuscar.setText("");
            this.jTFBuscar.requestFocus();
        }else{
            this.jTFBuscar.setEnabled(false);
            this.jLPrestamo.setText(" ENCONTRADO");
            jTPrestamo.setModel(tablaPrestamo(prestamo));
            if(jCBconsultar.getSelectedItemAt()=="codigo_l"){
                Libro libro=new Libro();
                libro=(Libro)
                libro.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFBuscar.getText() ));
                libro.actualizarDatos(libro.getCopias(), this.jTFcodigoL.getText());
                this.jTLibro.setModel(tablaLibro(libro));
                this.jBDisponibilidad.setVisible(true);
            }
        }
    }
}

}else{
    prestamo = (Prestamo)
    prestamo.buscarStringPrestamo(jCBconsultar.getSelectedItemAt().toString(),this.jTFBuscar.ge
    tText());

    if (prestamo == null) {
        JOptionPane.showMessageDialog(null, "No se ha encontrado el préstamo");
        this.jTFBuscar.setText(null);
        this.jTFBuscar.requestFocus();
    } else {
        this.jTFBuscar.setEnabled(false);
        this.jLPrestamo.setText(" ENCONTRADO");
        jTPrestamo.setModel(tablaPrestamo(prestamo));
    }
}

```

```
}
```

Disponibilidad

```
Libro libro=new Libro();
```

```
this.jPDisponibilidad.setVisible(true);
```

```
libro = (Libro)
```

```
libro.buscarIntLibro(jCBconsultar.getSelectedItem().toString(),Integer.parseInt(this.jTFBuscar.getText()));
```

```
if (libro == null) {
```

```
    JOptionPane.showMessageDialog(null, "No se ha encontrado el libro");
```

```
this.jTFBuscar.setText(null);
```

```
this.jTFBuscar.requestFocus();
```

```
} else {
```

```
this.jTFBuscar.setEnabled(false);
```

```
this.jLPrestamo.setText("ENCONTRADO");
```

```
jTLibro.setModel(tablaLibro(libro));
```

```
}
```

ENTREGAR

En este table es para realizar la entrega de un libro, el cual nos pide el codigo del estudiante, el codigo del libro, y la fecha de entrega.

```
Prestamo prestamo = new Prestamo();
```

```
prestamo = (Prestamo) prestamo.buscarPrestamo("codigo_e","codigo_l",
```

```
Integer.parseInt(this.jTFcodEentrega.getText()),
```

```
Integer.parseInt(this.jTFcodigoLe.getText()));
```

```
if (prestamo == null) {
```

```
    JOptionPane.showMessageDialog(null, "No se ha encontrado el préstamo");
```

```
this.jTFcodEentrega.setText(null);
```

```
this.jTFcodigoLe.setText(null);
```

```
this.jTFcodEentrega.requestFocus();
```

```
} else {
```



```

String estado="\Entregado\";

String sql ="UPDATE estudiantes_libros SET
estado="+estado+",fecha_e=\""+this.jTFfechaE.getText()+"\""+ " WHERE
codigo_e="+this.jTFcodEentrega.getText()+" AND codigo_l="+this.jTFcodigoLe.getText()
;

try{
    PreparedStatement pps= cn.prepareStatement(sql);
    pps.executeUpdate();

    Prestamo prestamo2= new Prestamo();

    prestamo2= (Prestamo) prestamo.buscarPrestamo("codigo_e","codigo_l",
    Integer.parseInt(this.jTFcodEentrega.getText()),
    Integer.parseInt(this.jTFcodigoLe.getText()));

    Libro libro2 = new Libro();

    libro2 =(Libro)
libro2.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoLe.getText()));

    int num=libro2.getCopias()+1;

    try{
        String sql1 = "UPDATE libros SET copias_l="+num + " WHERE
codigo_l="+this.jTFcodigoLe.getText();

        PreparedStatement ppsp= cn.prepareStatement(sql1);
        ppsp.executeUpdate();

        Libro libro = new Libro();

        libro =(Libro)
libro.buscarIntLibro("codigo_l",Integer.parseInt(this.jTFcodigoLe.getText()));

        this.jTLibro.setModel(tablaLibro(libro));

        this.jPDisponibilidad.setVisible(true);

    }catch(SQLException ex){
        Logger.getLogger(BDLibros.class.getName()).log(Level.SEVERE,null,ex);
        JOptionPane.showMessageDialog(null, ex);
        JOptionPane.showMessageDialog(null,"Ocurrio un error al intentar
Actualizar.");
    }

    this.jLPrestamo.setText("ENTREGADO");
    this.jTPrestamo.setModel(tablaPrestamo(prestamo2));

```

```

        eliminarTablaPrestamo(jTListaPrestamo);
        listarPrestamos();
    } catch (SQLException ex) {
        Logger.getLogger(BDEstudiantes.class.getName()).log(Level.SEVERE, null, ex);
        JOptionPane.showMessageDialog(null, ex);
        JOptionPane.showMessageDialog(null, "Ocurrio un error al intentar Actualizar.");
    }
}

```

Private void jTFcodEntregaActionPerformed

```

try {
    String valueToCheck = this.jTFcodEentrega.getText();
    String sql = "SELECT COUNT(*) FROM estudiantes WHERE codigo_e = ?";
    PreparedStatement stmt = cn.prepareStatement(sql);
    stmt.setString(1, valueToCheck);
    ResultSet rs = stmt.executeQuery();

    // Comprueba si hay al menos una fila que coincida con el valor especificado
    boolean valueExists = rs.next() && rs.getInt(1) > 0;

    if (valueExists) {
        System.out.println("El valor está presente en la tabla");
        this.jTFcodigoLe.requestFocus();
    } else {
        System.out.println("El valor no está presente en la tabla");
        JOptionPane.showMessageDialog(null, "Código no existe en la lista.\nIngrese un nuevo código ");
        this.jTFcodEentrega.setText(null);
        this.jTFcodEentrega.requestFocus();
    }

} catch (SQLException e) {

```

```

        System.out.println(e.getMessage());
    }

```

FILTRAR

Esta funcion nos permite a mostrar los prestamos con características similares, para evitar confusiones en el momento de verificar datos, como prestamos actuales, copias actuales de libros prestados, etc.

```

JDTablaPrestamo jdtablaprestamo=new JDTablaPrestamo();

jdtablaprestamo.setVisible(true);

```

Codigos Complementarios

```

public DefaultTableModel tablaPrestamo(Prestamo prestamo){
    DefaultTableModel modelo = (DefaultTableModel) jTPrestamo.getModel();
    int a = jTPrestamo.getRowCount()-1;
    for (int i = a; i >= 0; i--) {
        modelo.removeRow(modelo.getRowCount()-1);
    }
    String[] fila=new String[5];
        fila[0]=String.valueOf(prestamo.getCodigo_e());
        fila[1]=String.valueOf(prestamo.getCodigo_l());
        fila[2]=prestamo.getFechaPrestamo();
        fila[3]=prestamo.getEstado();
        fila[4]=prestamo.getFechaEntrega();
        modelo.addRow(fila);
    return modelo;
}

```

```

public void validarNumeros( java.awt.event.KeyEvent evt,javax.swing.JTextField textField
){
    char variable=evt.getKeyChar();
    if(Character.isLetter(variable)){
        getToolkit().beep();
    }
}

```

```

        evt.consume();

        JOptionPane.showMessageDialog(null, " Ingrese números");
        textField.requestFocus();
    }
}

public void listarPrestamos(){
    String sql="SELECT * FROM estudiantes_libros";
    Statement st;
    DefaultTableModel modelo2 = (DefaultTableModel) jTableListaPrestamo.getModel();

    String[] dato=new String[5];
    try{
        st=cn.createStatement();
        ResultSet result;
        result = st.executeQuery(sql);
        while(result.next()){
            dato[0]=result.getString(1);
            dato[1]=result.getString(2);
            dato[2]=result.getString(3);
            dato[3]=result.getString(4);
            dato[4]=result.getString(5);
            modelo2.addRow(dato);
        }
        jTableListaPrestamo.setModel(modelo2);
    }catch(SQLException e){
        e.printStackTrace();
    }
}

public void eliminarTablaPrestamo(javax.swing.JTable Jtable){
    DefaultTableModel tb = (DefaultTableModel) Jtable.getModel();
    int a = Jtable.getRowCount()-1;

```

```

        for (int i = a; i >= 0; i--) {
            tb.removeRow(tb.getRowCount()-1);
        }
        //cargaTicket();
    }

    public DefaultTableModel tablaLibro(Libro libro){
        DefaultTableModel modelo = (DefaultTableModel) jTLibro.getModel();
        int a = jTLibro.getRowCount()-1;
        for (int i = a; i >= 0; i--) {
            modelo.removeRow(modelo.getRowCount()-1);
        }
        String[] fila=new String[5];
        fila[0]=String.valueOf(libro.getCodigo());
        fila[1]=libro.getTitulo();
        fila[2]=libro.getAutor();
        fila[3]=String.valueOf(libro.getCopias());
        fila[4]=String.valueOf(libro.getTotal());
        modelo.addRow(fila);
        return modelo;
    }

```

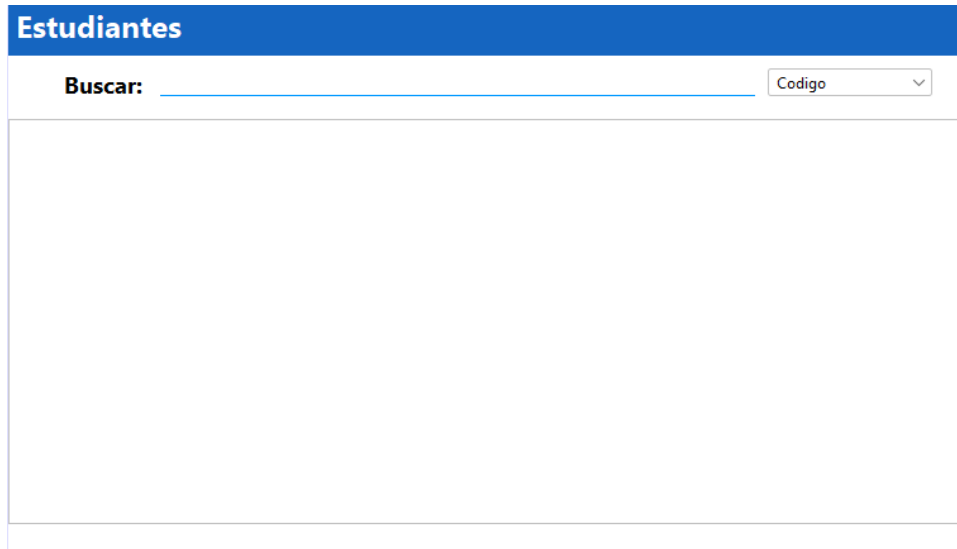
Para la validacion del codigo del libro o del estudiante, se utiliza el siguiente codigo.

```

validarNumeros(evt,this.jTFcodigoL)
validarNumeros(evt,this.jTFcodigoE);

```

JDtablaEstudiantes:



En este JFrame se utilizó un combo box junto con el JTable y un JTextField como barra de búsqueda, y se creó un evento KeyReleased en el JTextField para que cada vez que se escriba algo dentro de la barra de búsqueda se ejecute el método buscar().

```
private void JTFbusquedaKeyReleased(java.awt.event.KeyEvent evt) {  
    buscar(this.JTFbusqueda.getText());  
}
```

```
public void buscar(String buscar){  
    modelo.setRowCount(0);  
    String sql = null;  
    switch(this.JCBbuscar.getSelectedIndex()){  
        case 0:  
            sql="SELECT * FROM estudiantes WHERE codigo_e LIKE '%" + buscar + "%'";  
            break;  
        case 1:  
            sql="SELECT * FROM estudiantes WHERE nombre_e LIKE '%" + buscar + "%'";  
            break;  
        case 2:  
            sql="SELECT * FROM estudiantes WHERE apellido_e LIKE '%" + buscar + "%'";  
            break;
```

```

        case 3:
            sql="SELECT * FROM estudiantes WHERE fechaNacimiento_e LIKE
'%" +buscar+"%";

            break;

        case 4:
            sql="SELECT * FROM estudiantes WHERE cedula_e LIKE '%" +buscar+"%";

            break;

        case 5:
            sql="SELECT * FROM estudiantes WHERE direccion LIKE '%" +buscar+"%";

            break;

        case 6:
            sql="SELECT * FROM estudiantes WHERE telefono LIKE '%" +buscar+"%";

            break;
    }

```

```

String datos[]=new String[7];

try {
    rs = con.createStatement().executeQuery(sql);
    while(rs.next()==true){
        datos[0]=rs.getString(1);
        datos[1]=rs.getString(2);
        datos[2]=rs.getString(3);
        datos[3]=rs.getString(4);
        datos[4]=rs.getString(5);
        datos[5]=rs.getString(6);
        datos[6]=rs.getString(7);
        modelo.addRow(datos);
    }
    jTablela.setModel(modelo);
} catch (SQLException ex) {

    Logger.getLogger(JDtablaEstudiantes.class.getName()).log(Level.SEVERE, null,
ex);

```



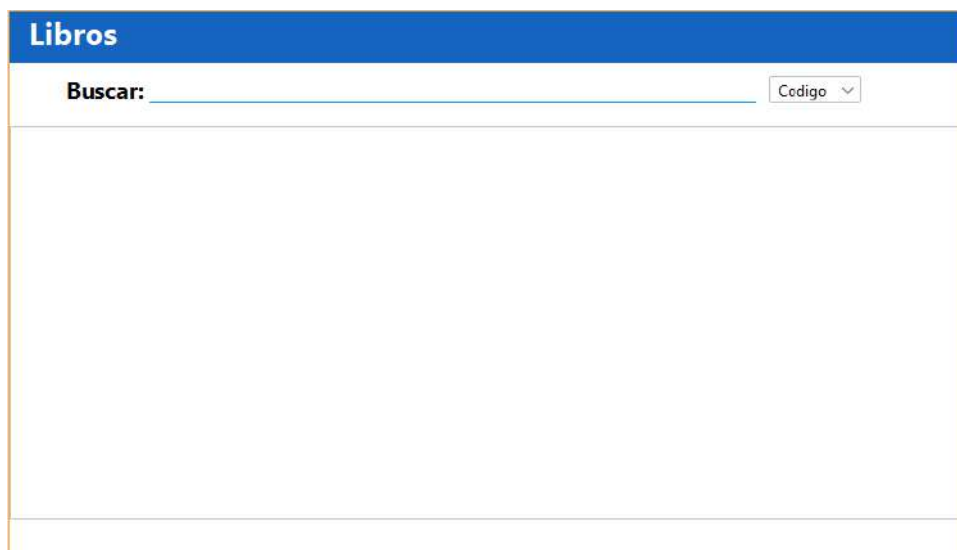
```

    }
}

```

Este método lo que hace es que le llega un String que sería lo que está escrito en la barra de búsqueda para posteriormente ingresar a la base de datos y según lo que escojamos en el ComboBox (en este caso las opciones son: Código, Nombres, Apellidos, Fecha Nacimiento, Cedula, Dirección, Teléfono) busca el apartado dentro de la Base de Datos donde contenga el mismo String escrito en la barra de búsqueda, toma la fila donde se encuentra este apartado, la copia e imprime en la JTable que tenemos, para luego pasar con el siguiente apartado que también contenga este String, y ya que esto se va ejecutando cada vez que ingresamos algo en el JTextField (debido al KeyReleased) la tabla se actualiza mientras escribimos.

JDtablaLibros:



En este JFrame se utilizó la misma estructura y lógica que el anterior, solamente modificando el método buscar() ya que los datos que poseen los libros son diferentes.

```

private void JTFbusquedaKeyReleased(java.awt.event.KeyEvent evt) {
    buscar(this.JTFbusqueda.getText());
}

```

```

public void buscar(String buscar){
    modelo.setRowCount(0);
    String sql = null;
    switch(this.JCBbusqueda.getSelectedIndex()){
        case 0:
            sql="SELECT * FROM libro WHERE codigo_1 LIKE '%" + buscar + "%'";

```

```

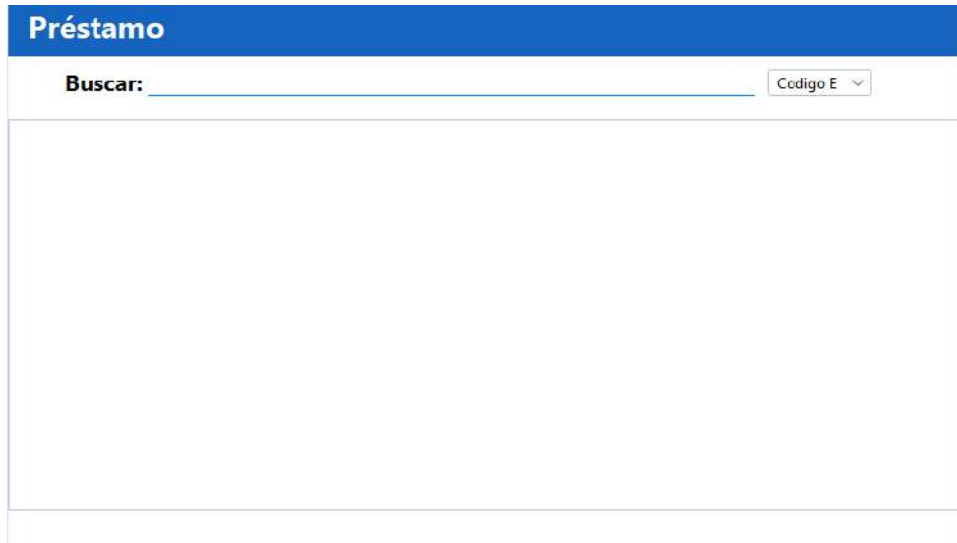
        break;
    case 1:
        sql="SELECT * FROM libro WHERE titulo_1 LIKE '%" + buscar + "%'";
        break;
    case 2:
        sql="SELECT * FROM libro WHERE autor_1 LIKE '%" + buscar + "%'";
        break;

}

String datos[]=new String[5];
try {
    rs = con.createStatement().executeQuery(sql);
    while(rs.next()==true){
        datos[0]=rs.getString(1);
        datos[1]=rs.getString(2);
        datos[2]=rs.getString(3);
        datos[3]=rs.getString(4);
        datos[4]=rs.getString(5);
        modelo.addRow(datos);
    }
    jTable1.setModel(modelo);
} catch (SQLException ex) {
    Logger.getLogger(JDtablaLibros.class.getName()).log(Level.SEVERE, null, ex);
}
}

```

JDtablaPréstamo:



Se utilizó la misma estructura y lógica pero cambiando el método buscar().

```
private void JTFbusquedaKeyReleased(java.awt.event.KeyEvent evt) {  
    buscar(this.JTFbusqueda.getText());  
}  
  
public void buscar(String buscar){  
    modelo.setRowCount(0);  
    String sql = null;  
    switch(this.JCBbusqueda.getSelectedIndex()){  
        case 0:  
            sql="SELECT * FROM estudiantes_libros WHERE codigo_e LIKE  
'%"+buscar+"%";  
            break;  
        case 1:  
            sql="SELECT * FROM estudiantes_libros WHERE codigo_l LIKE  
'%"+buscar+"%";  
            break;  
        case 2:  
            sql="SELECT * FROM estudiantes_libros WHERE fecha_p LIKE  
'%"+buscar+"%";  
            break;
```

```
}
```

```
String datos[]=new String[5];
```

```
try {
```

```
    rs = con.createStatement().executeQuery(sql);
```

```
    while(rs.next()==true){
```

```
        datos[0]=rs.getString(1);
```

```
        datos[1]=rs.getString(2);
```

```
        datos[2]=rs.getString(3);
```

```
        datos[3]=rs.getString(4);
```

```
        datos[4]=rs.getString(5);
```

```
        modelo.addRow(datos);
```

```
    }
```

```
    jTable1.setModel(modelo);
```

```
} catch (SQLException ex) {
```

```
    Logger.getLogger(JDtablaPrestamo.class.getName()).log(Level.SEVERE, null, ex);
```

```
}
```

```
}
```

RESULTADOS

Ejecución del programa:

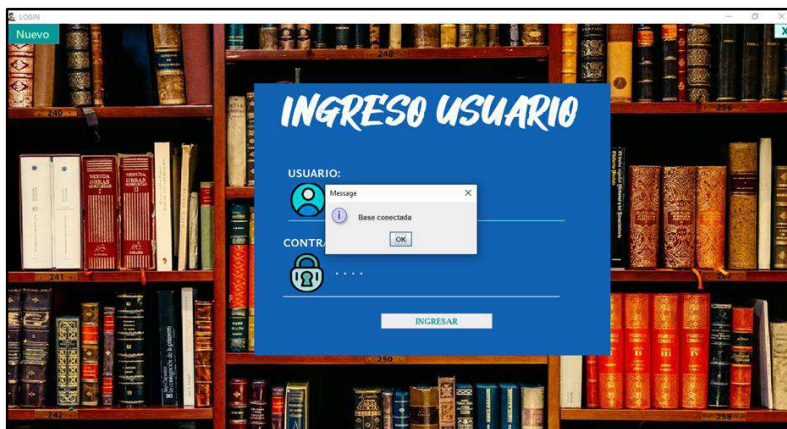
1.-Ingreso de credenciales



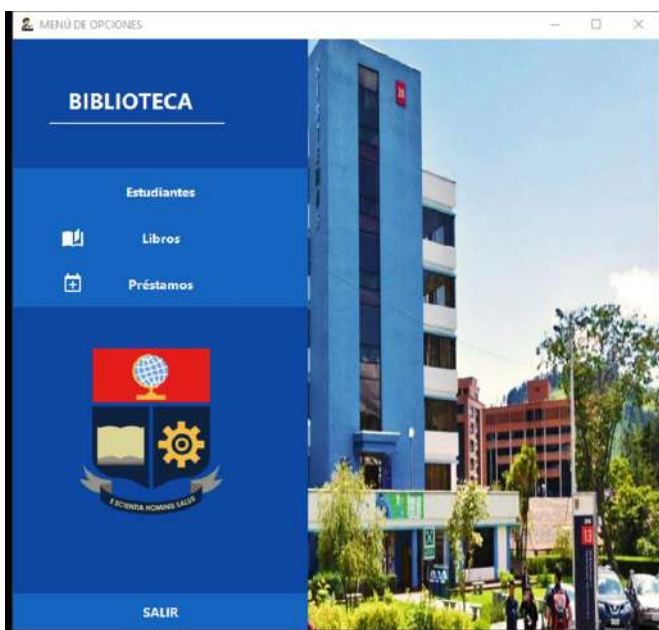
El login presenta en la parte superior izquierda la opción de nuevo y en la parte derecha la opción de salir. Si el ingreso es incorrecto:



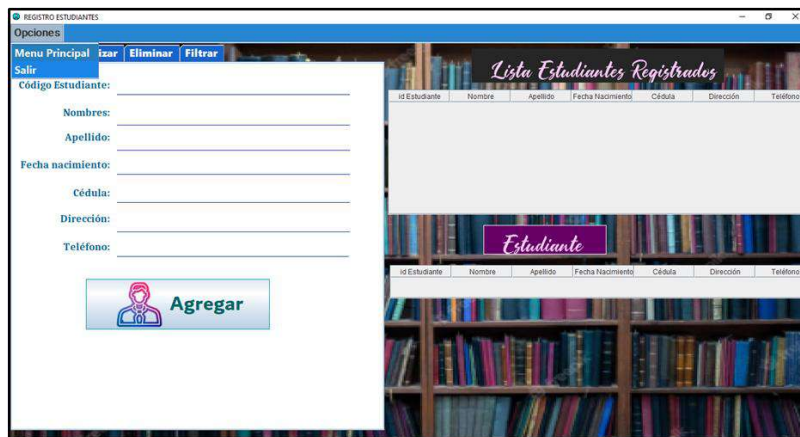
Caso contrario:



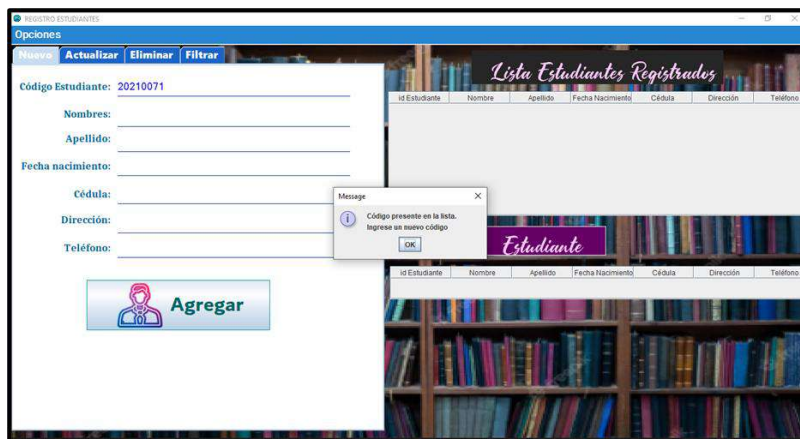
Seguidamente se despliega el menú con las opciones de estudiantes, libros, préstamos y salir:



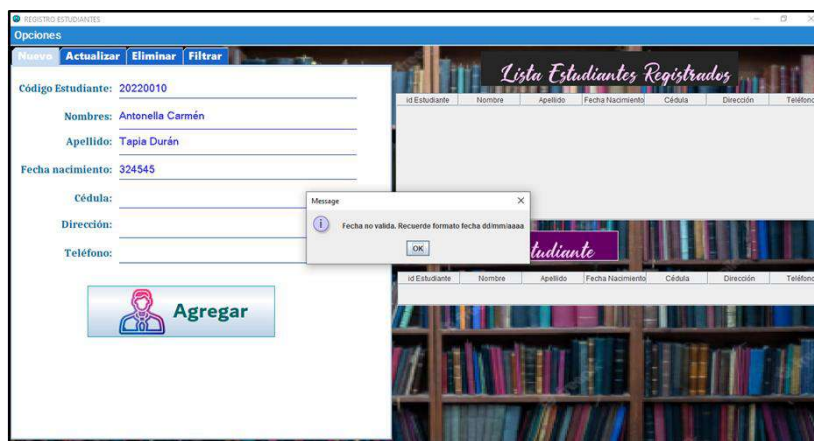
Opción estudiantes (el usuario puede ingresar, actualizar, eliminar y filtrar los estudiantes) en la parte superior izquierda siempre estará disponible regresar al menú principal y salir, mientras el usuario realice un cambio de panel del jTable los atributos regresan a un estado de “nuevo” (se colocan espacios en blanco) :



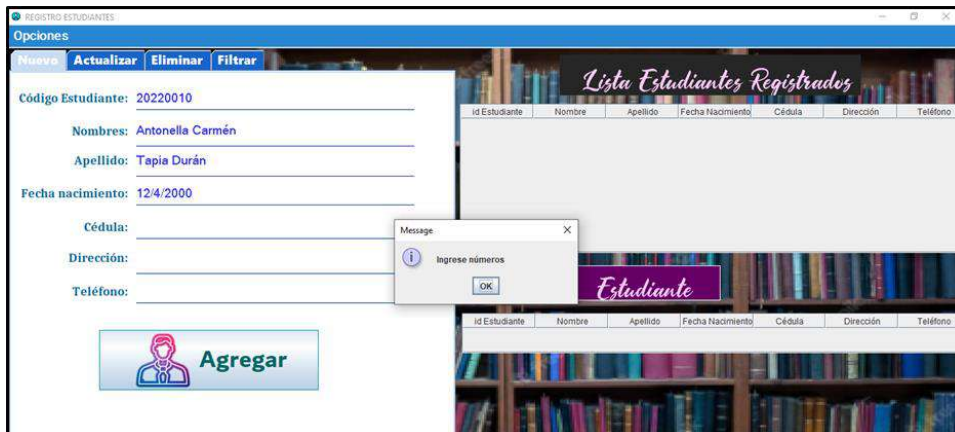
Opción estudiantes Ingreso, si el código que trata de ingresar ya existe (recordar que el código es único).



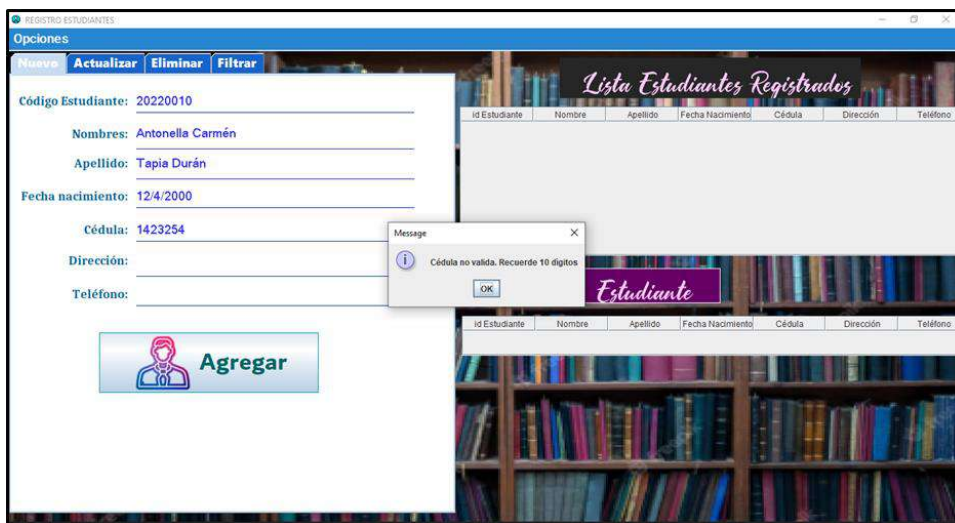
Opción estudiantes Ingreso, fecha fuera del formato indicado:



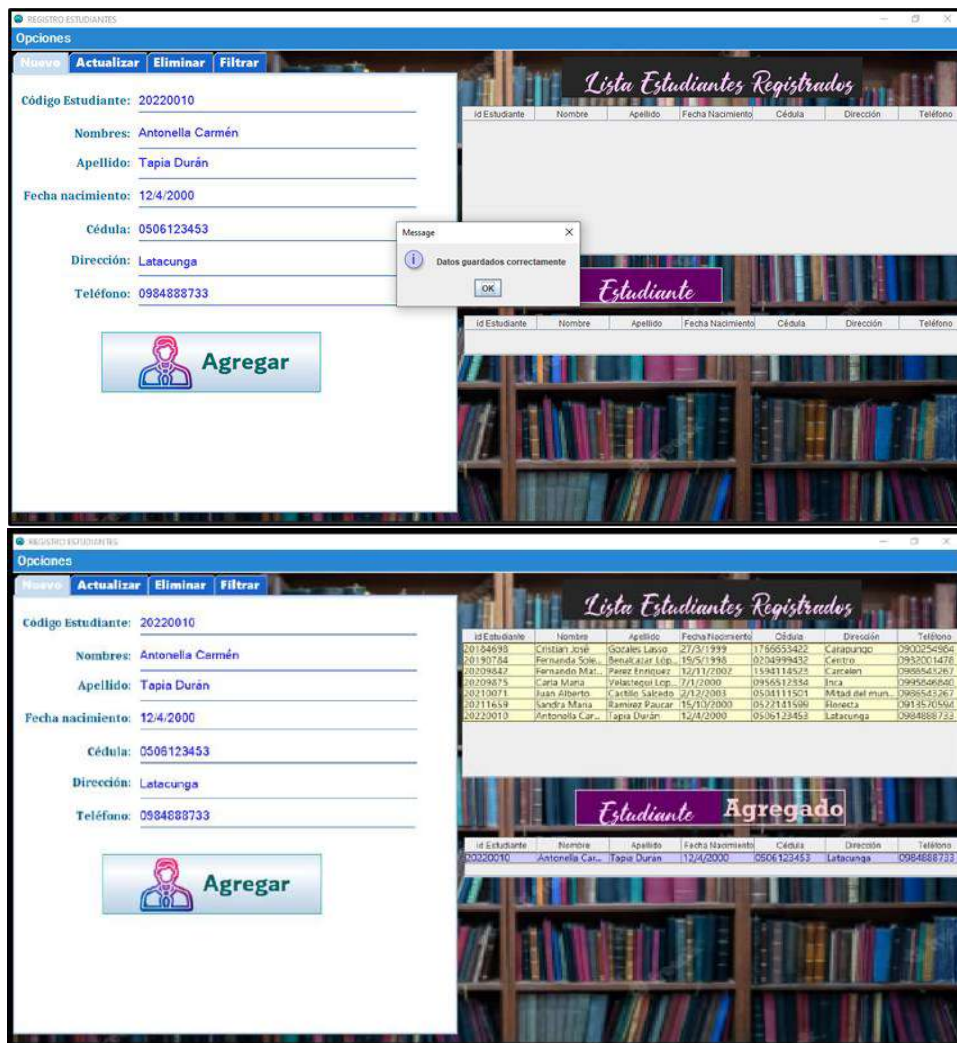
Opción estudiantes Ingreso, cédula ingreso incorrecto letras, para teléfono se repite el proceso:



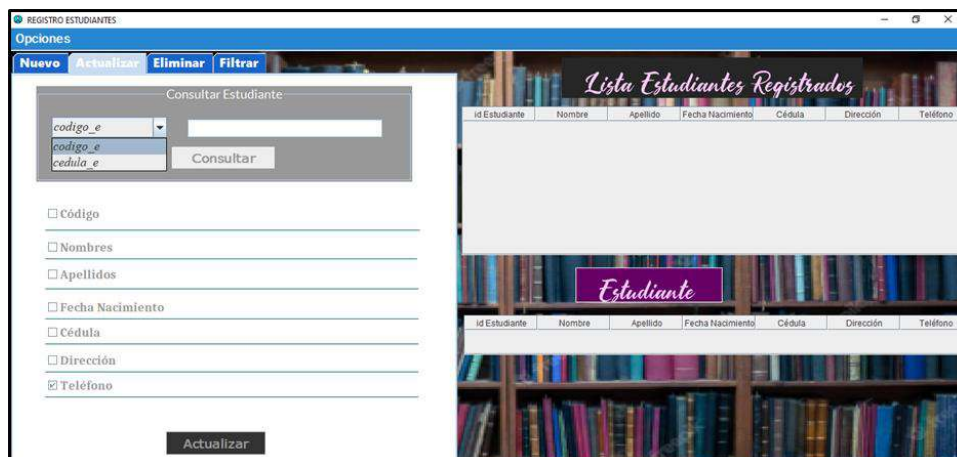
Opción estudiantes Ingreso, cédula incorrecta (número de dígitos insuficientes) para teléfono se repite el proceso:



Opción estudiantes Ingreso, ingreso correcto:



Opción estudiantes Actualizar, lo primero es consultar para ello se elige un parámetro (se tiene a disposición codigo_e,cedula_e pues son únicos para cada estudiante).



Una vez elegido el parámetro se habilita la caja de texto para poder ingresar el valor a buscar y se despliega la lista de estudiantes registrados:

REGISTRO ESTUDIANTES

Opciones: **Nuevo** **Actualizar** **Eliminar** **Filtrar**

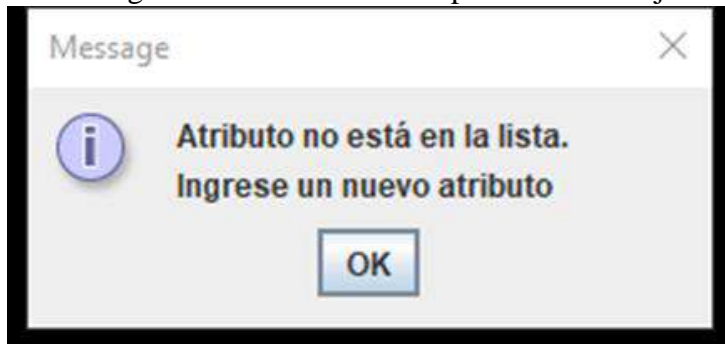
Consultar Estudiante:

☐ Código
☐ Nombres
☐ Apellidos
☐ Fecha Nacimiento
☐ Cédula
☐ Dirección
☒ Teléfono

Lista Estudiantes Registrados

id Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20184698	Cristian José	Gonzales Lasso	27/3/1999	1766653422	Carapungo	0900254984
20190784	Fernanda Sol...	Bernalcar Lóp...	19/5/1998	0204999432	Centro	0933201478
20209842	Fernando Mat...	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267
20209875	Carla Maria	Velastegu Lóp...	7/1/2000	0956512334	Inca	0995846840
20210071	Juan Alberto	Castillo Salcedo	27/12/2003	0504111501	Mitad del mun...	0986543267
20211659	Sandra Maria	Ramirez Paucar	15/10/2000	0522141599	Floresta	0913570594
20220010	Antonella Car...	Tapia Durán	12/4/2000	0506123453	Latacunga	0984888733

Si el código o la cédula no existe aparece el mensaje:



Por el contrario, aparece el estudiante con sus datos y un mensaje indicando que debe seleccionar los atributos si los desea cambiar.

REGISTRO ESTUDIANTES

Opciones: **Nuevo** **Actualizar** **Eliminar** **Filtrar**

Consultar Estudiante:

☐ Código **20220010**
☐ Nombres **Antonella Carmén**
☐ Apellidos **Tapia Durán**
☐ Fecha Nacimiento **12/4/2000**
☐ Cédula **0506123453**
☐ Dirección **Latacunga**
☒ Teléfono **0984888733**

Lista Estudiantes Registrados

id Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20184698	Cristian José	Gonzales Lasso	27/3/1999	1766653422	Carapungo	0900254984
20190784	Fernanda Sol...	Bernalcar Lóp...	19/5/1998	0204999432	Centro	0933201478
20209842	Fernando Mat...	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267
20209875	Carla Maria	Velastegu Lóp...	7/1/2000	0956512334	Inca	0995846840
20210071	Juan Alberto	Castillo Salcedo	27/12/2003	0504111501	Mitad del mun...	0986543267
20211659	Sandra Maria	Ramirez Paucar	15/10/2000	0522141599	Floresta	0913570594
20220010	Antonella Car...	Tapia Durán	12/4/2000	0506123453	Latacunga	0984888733

Message: Seleccione el o los elementos a modificar

Estudiante ENCONTRADO

id Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20220010	Antonella Car...	Tapia Durán	12/4/2000	0506123453	Latacunga	0984888733

Si no selecciona ningún atributo y coloca actualizar:

REGISTRO ESTUDIANTES

Opciones: Nuevo Actualizar Eliminar Filtrar

Consultar Estudiante

codigo_e: 20220010

Consultar

☐ Código: 20220010

☐ Nombres: Antonella Carmén

☐ Apellidos: Tapia Durán

☐ Fecha Nacimiento: 12/4/2000

☐ Cédula: 0506123453

☐ Dirección: Latacunga

☐ Teléfono: 0984888733

Actualizar

Lista Estudiantes Registrados

ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20184698	Cristian José	Gonzales Lasso	27/3/1999	1766653422	Carapungo	0900254984
20190784	Fernanda Sol...	Bernalcazar Lop...	19/5/1998	0204999432	Centro	0932001478
20209842	Fernando Mat...	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267
20209875	Carla Maria	Velastegui Lop...	7/1/2000	0956512334	Inca	0995846840
20210071	Juan Alberto	Castillo Salcedo	2/12/2003	0504111501	Mitad del mun...	0986543267
20211659	Sandra Maria	Ramirez Pausar	15/10/2000	0522141599	Ploresta	0913570594
20220010	Antonella Car...	Tapia Durán	12/4/2000	0506123453	Latacunga	0984888733

Message: No se puede actualizar porque no hubo cambio

Estudiante ENCONTRADO

ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20220010	Antonella Car...	Tapia Durán	12/4/2000	0506123453	Latacunga	0984888733

Dada una actualización correcta la pantalla es:

REGISTRO ESTUDIANTES

Opciones: Nuevo Actualizar Eliminar Filtrar

Consultar Estudiante

codigo_e: 20220010

Consultar

☐ Código: 20220010

☐ Nombres: Antonella Carmén

☒ Apellidos: Tapia Campos

☐ Fecha Nacimiento: 12/4/2000

☐ Cédula: 0506123453

☒ Dirección: La Maná

☐ Teléfono: 0984888733

Actualizar

Message: Datos Actualizados correctamente

Lista Estudiantes Registrados

ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20184698	Cristian José	Gonzales Lasso	27/3/1999	1766653422	Carapungo	0900254984
20190784	Fernanda Sol...	Bernalcazar Lop...	19/5/1998	0204999432	Centro	0932001478
20209842	Fernando Mat...	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267
20209875	Carla Maria	Velastegui Lop...	7/1/2000	0956512334	Inca	0995846840
20210071	Juan Alberto	Castillo Salcedo	2/12/2003	0504111501	Mitad del mun...	0986543267
20211659	Sandra Maria	Ramirez Pausar	15/10/2000	0522141599	Ploresta	0913570594
20220010	Antonella Car...	Tapia Campos	12/4/2000	0506123453	La Maná	0984888733

Estudiante ENCONTRADO

ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20220010	Antonella Car...	Tapia Campos	12/4/2000	0506123453	La Maná	0984888733

Posteriormente se presenta los datos previos del estudiante y tanto la lista como el estudiante se actualiza:

REGISTRO ESTUDIANTES

Opciones: Nuevo Actualizar Eliminar Filtrar

Consultar Estudiante

codigo_e: 20220010

Consultar

☐ Código: 20220010

☐ Nombres: Antonella Carmén

☒ Apellidos: Tapia Campos

☐ Fecha Nacimiento: 12/4/2000

☐ Cédula: 0506123453

☒ Dirección: La Maná

☐ Teléfono: 0984888733

Actualizar

Lista Estudiantes Registrados

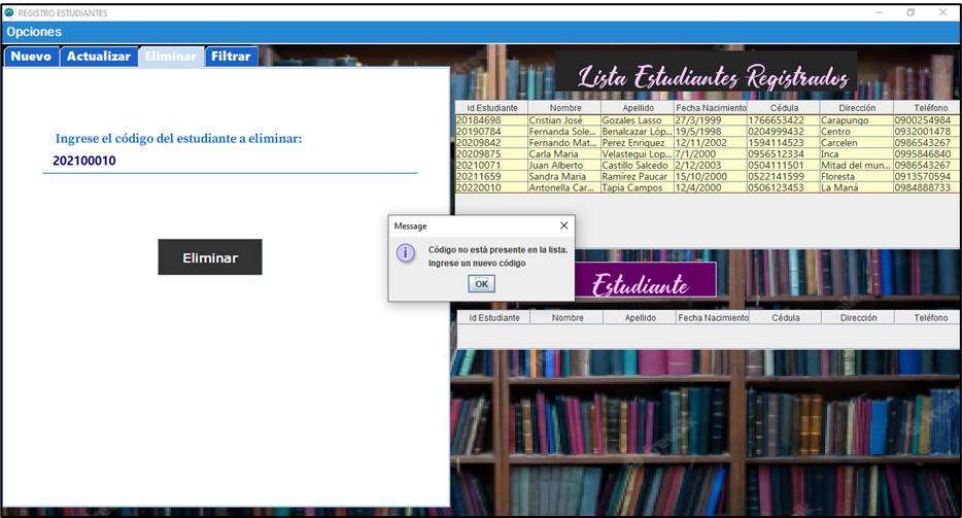
ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20184698	Cristian José	Gonzales Lasso	27/3/1999	1766653422	Carapungo	0900254984
20190784	Fernanda Sol...	Bernalcazar Lop...	19/5/1998	0204999432	Centro	0932001478
20209842	Fernando Mat...	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267
20209875	Carla Maria	Velastegui Lop...	7/1/2000	0956512334	Inca	0995846840
20210071	Juan Alberto	Castillo Salcedo	2/12/2003	0504111501	Mitad del mun...	0986543267
20211659	Sandra Maria	Ramirez Pausar	15/10/2000	0522141599	Ploresta	0913570594
20220010	Antonella Car...	Tapia Campos	12/4/2000	0506123453	La Maná	0984888733

Estudiante ENCONTRADO

Estudiante ACTUAL

ID Estudiante	Nombre	Apellido	Fecha Nacimiento	Cédula	Dirección	Teléfono
20220010	Antonella Car...	Tapia Campos	12/4/2000	0506123453	La Maná	0984888733

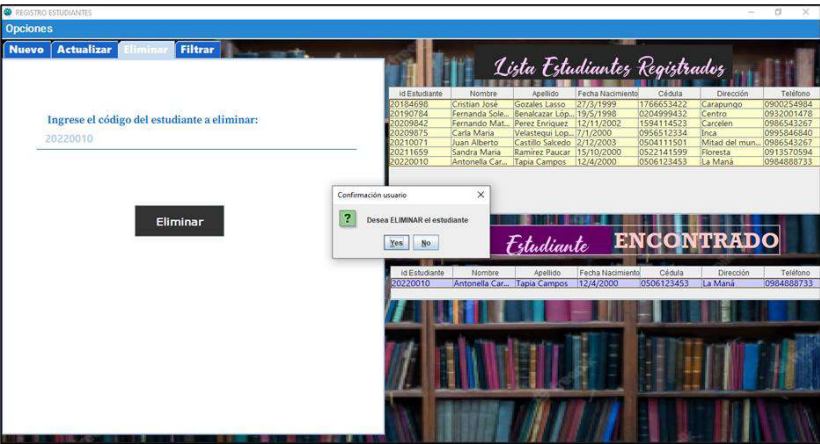
Opción estudiantes Eliminar, para eliminar considera el parámetro de código del estudiante pues es único, despliega la lista. Puede que el código no se registre por ello muestra un mensaje de error:



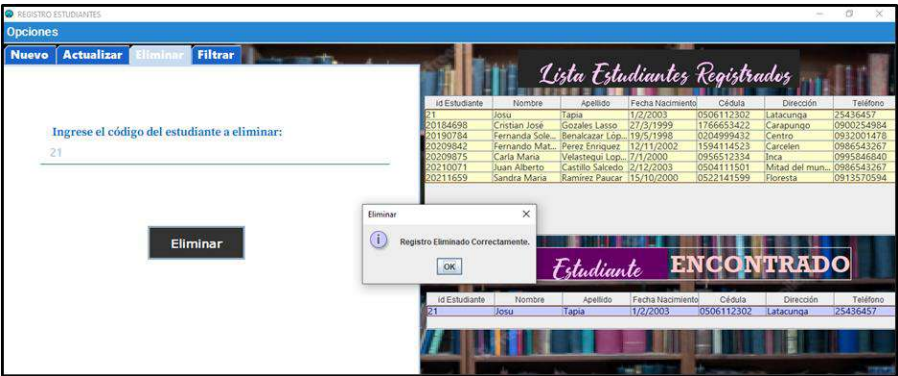
Si trata de ingresar letras:

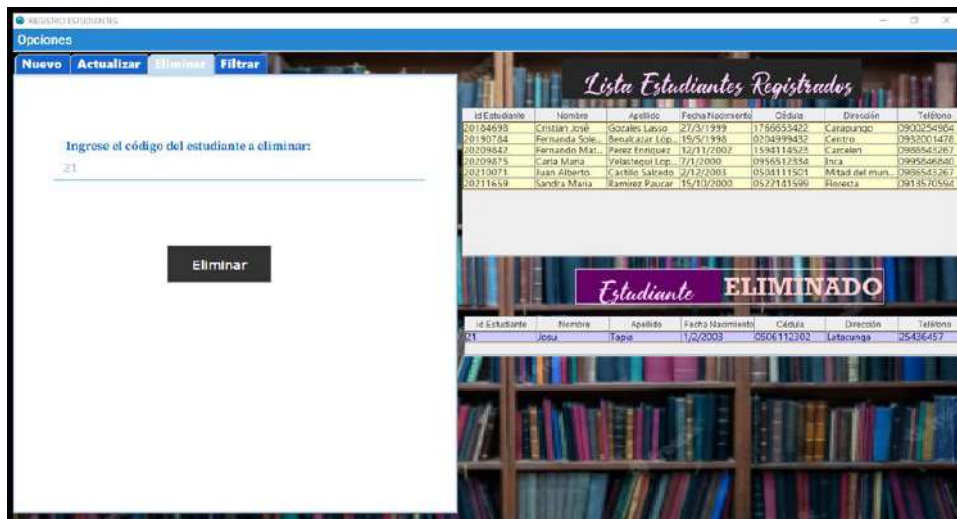


Si encuentra el estudiante muestra sus datos y pregunta si realmente desea eliminar:

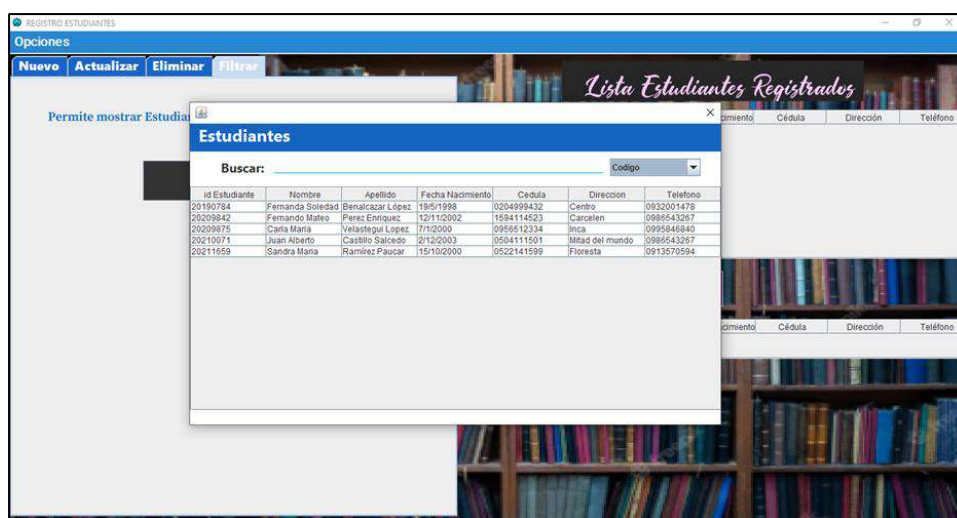
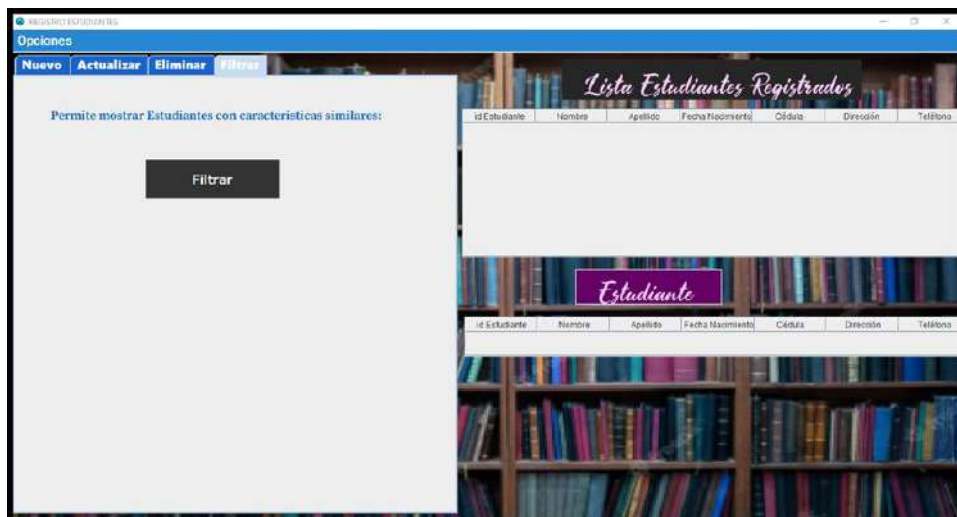


Finalmente, se elimina el registro:





La última pestaña es filtrar incluye un botón que despliega una tabla:



El filtro presenta búsqueda mediante atributos, para activar el textfield se debe elegir un parámetro de búsqueda:

id Estudiante	Nombre	Apellido	Fecha Nacimiento	Cedula	Direccion	Telefono
20190784	Fernanda Soledad	Benalcazar López	19/5/1998	0204999432	Centro	0932001478
20209842	Fernando Mateo	Perez Enriquez	12/11/2002	1594114523	Carcelen	0986543267

Aparecerán los registros que coincidan con ese atributo.

Opción libros(el usuario puede ingresar, actualizar, eliminar y filtrar los libros) en la parte superior izquierda siempre estará disponible regresar al menú principal y salir, mientras el usuario realice un cambio de panel del jTable los atributos regresan a un estado de “nuevo” (se colocan espacios en blanco) :

REGISTRO LIBROS
Opciones
Menu Principal
Salir

Código Libro:
Titulo:
Autor:
Número Copias:

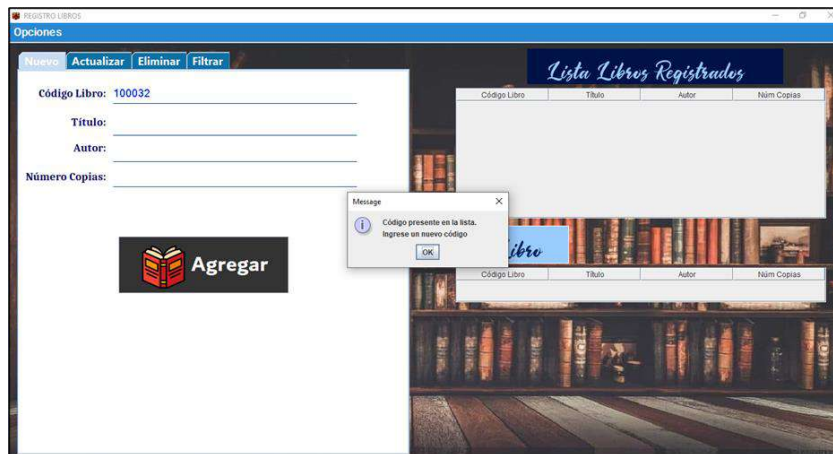
Lista Libros Registrados

Código Libro	Titulo	Autor	Num Copias

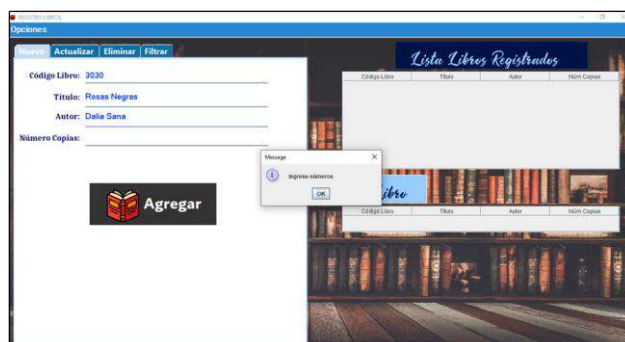
Libro

Código Libro	Titulo	Autor	Num Copias

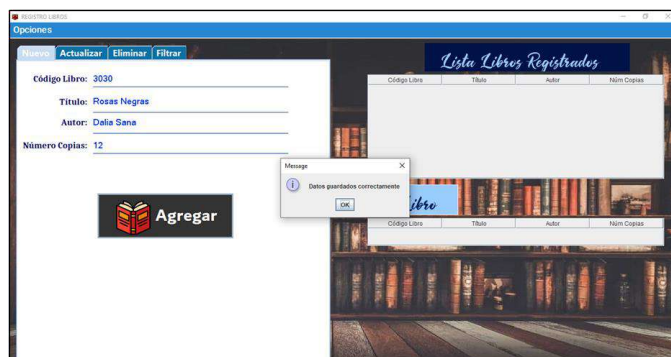
Opción libro Ingreso, si el código que trata de ingresar ya existe (recordar que el código es único).



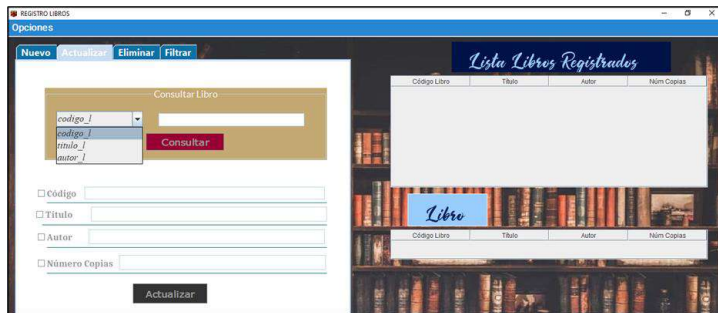
Opción Libros Ingreso, copias ingreso incorrecto letras:



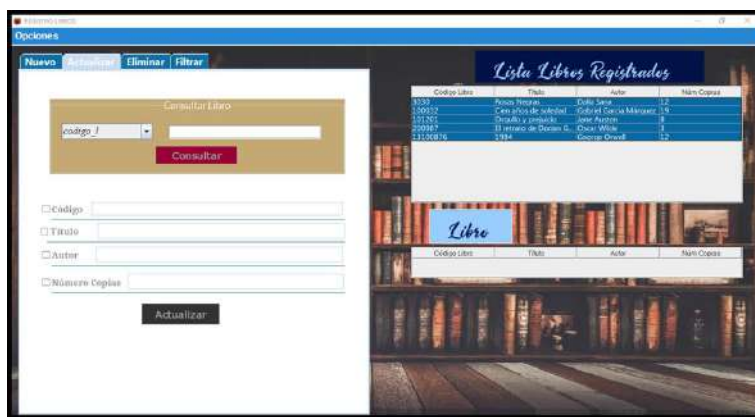
Opción Libros Ingreso, ingreso correcto:



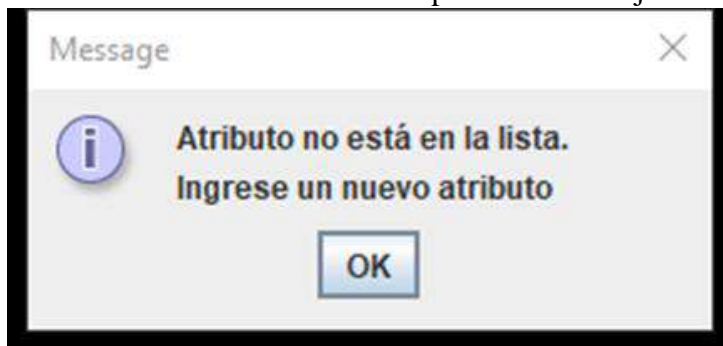
Opción Libros Actualizar, lo primero es consultar para ello se elige un parámetro (se tiene a disposición codigo_1, titulo_1).



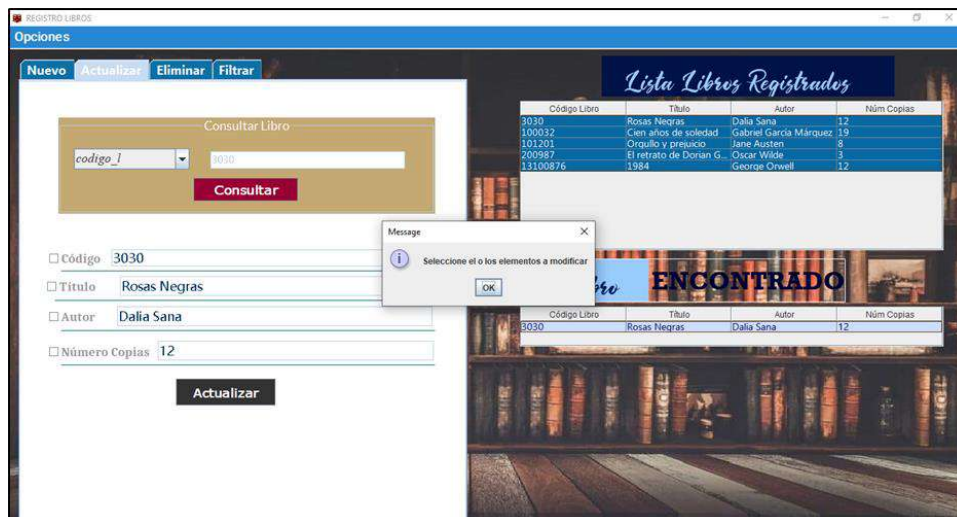
Una vez elegido el parámetro se habilita la caja de texto para poder ingresar el valor a buscar y se despliega la lista de estudiantes registrados:



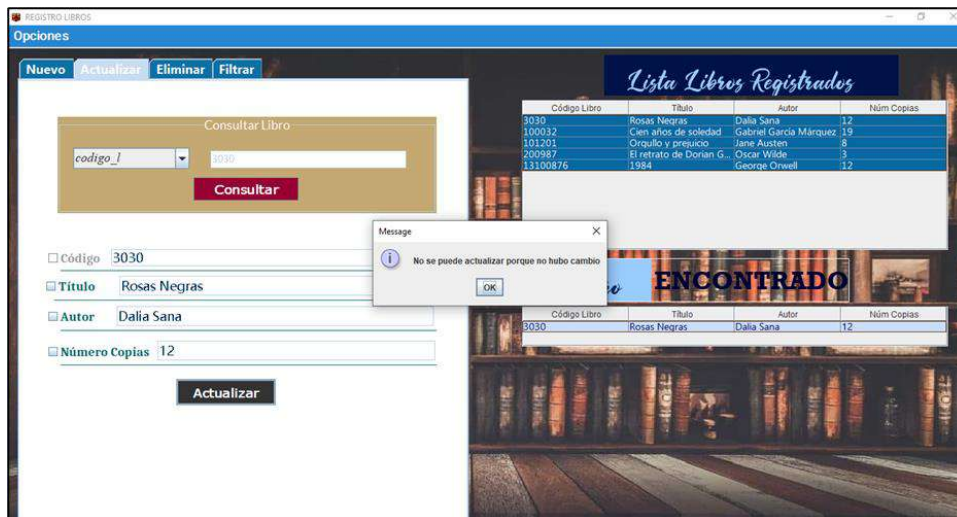
Si el atributo a buscar no existe aparece el mensaje:



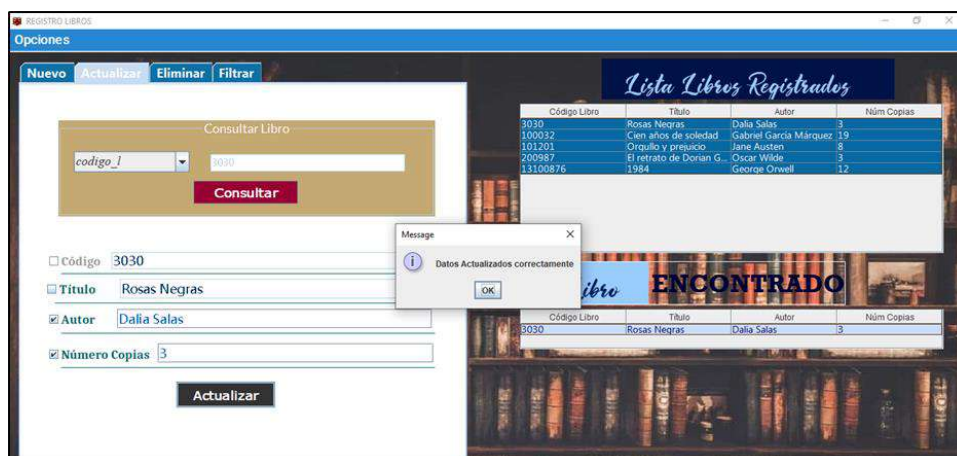
Por el contrario, aparece el libro con sus datos y un mensaje indicando que debe seleccionar los atributos si los desea cambiar.



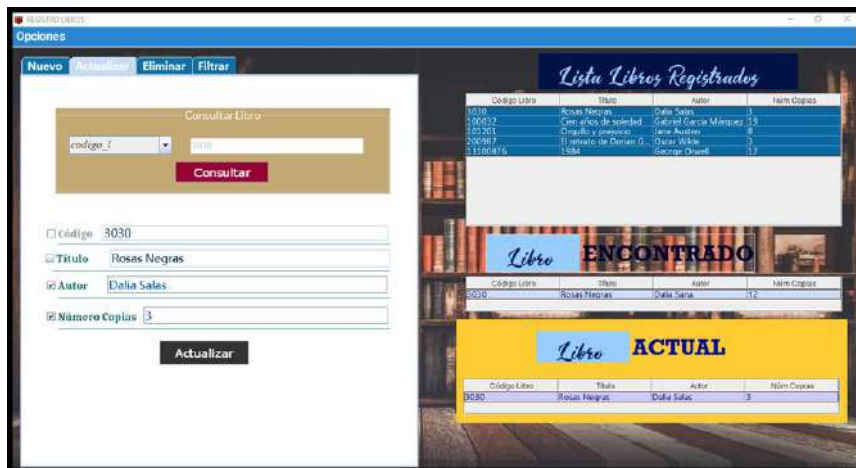
Si no selecciona ningún atributo y coloca actualizar:



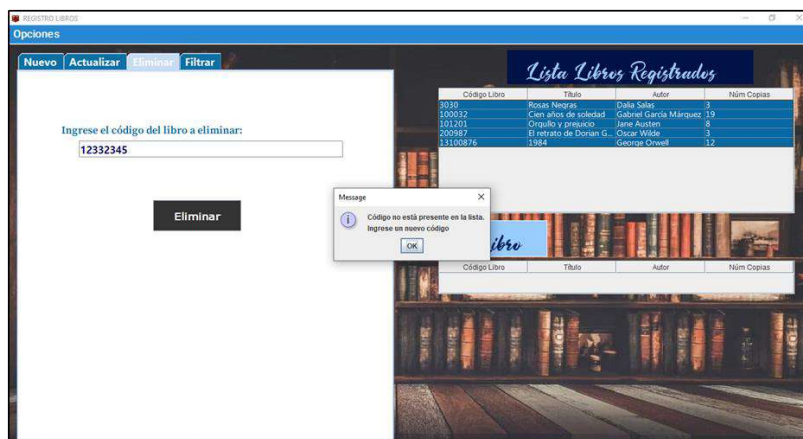
Dada una actualización correcta la pantalla es:



Posteriormente se presenta los datos previos del libro y tanto la lista como el libro se actualiza:

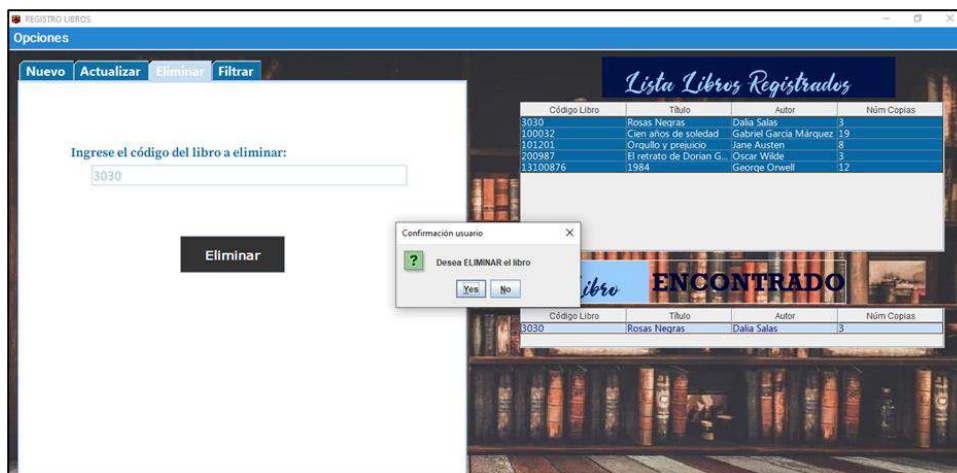


Opción Libros Eliminar, para eliminar considera el parámetro de código del libro pues es único, despliega la lista. Puede que el código no se registre por ello muestra un mensaje de error:

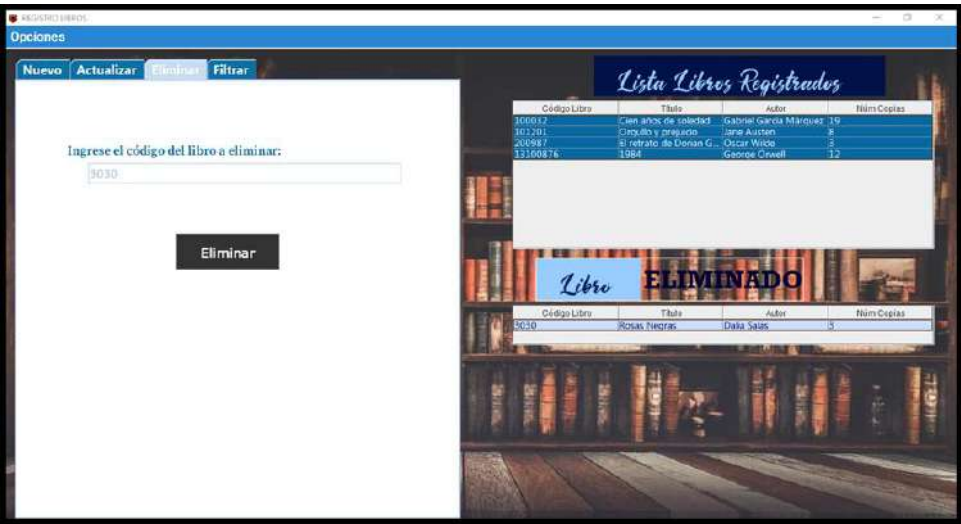
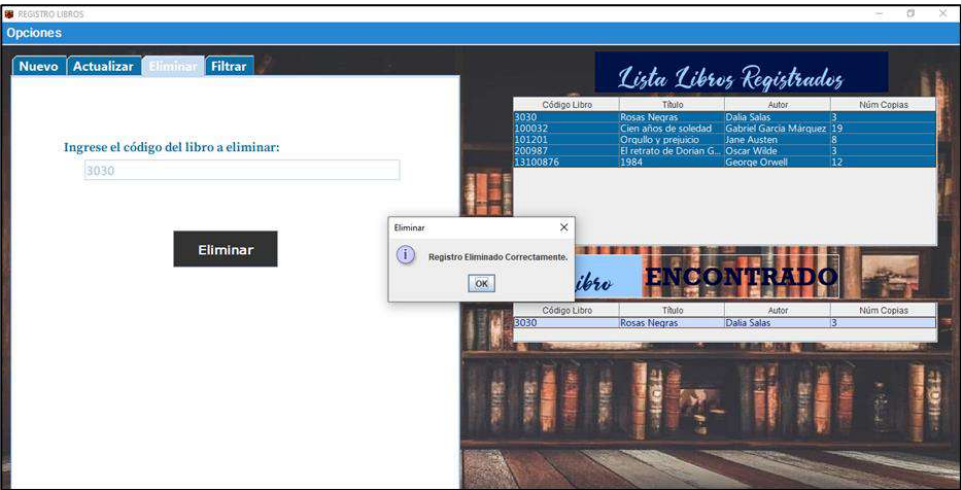


Si trata de ingresar letras:

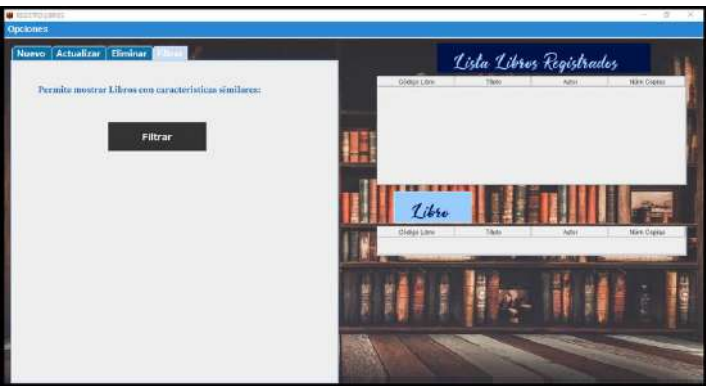
Si encuentra el libro muestra sus datos y pregunta si realmente desea eliminar:

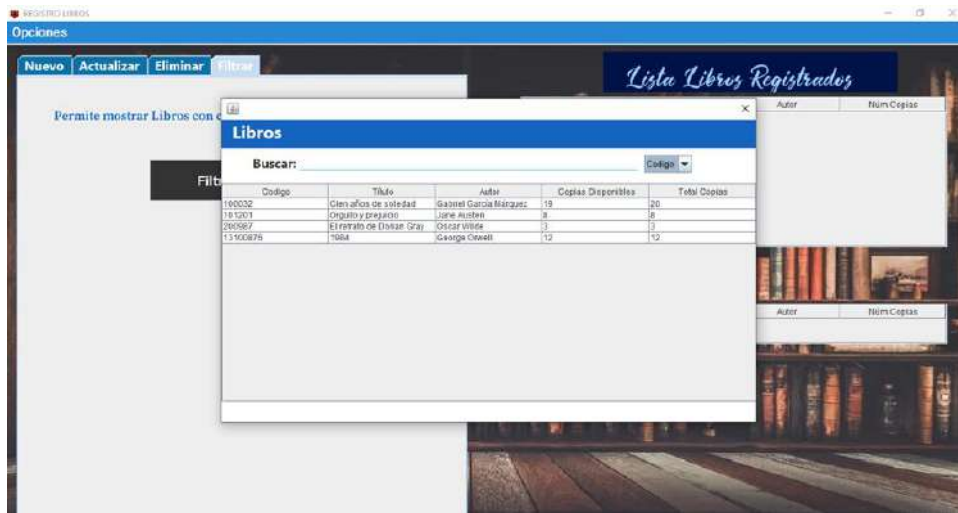


Finalmente, se elimina el registro:

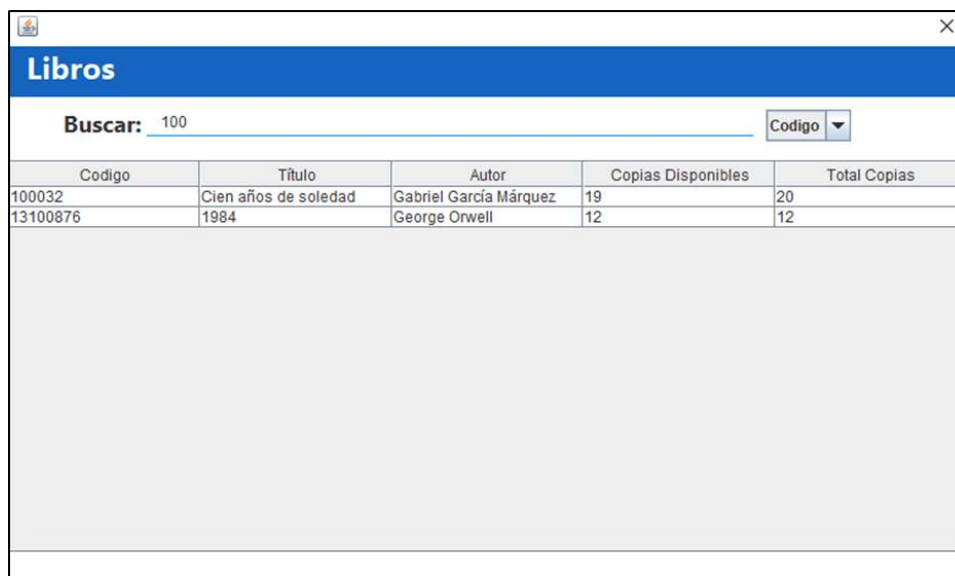


La última pestaña es filtrar incluye un botón que despliega una tabla:





El filtro presenta búsqueda mediante atributos, para activar el textfield se debe elegir un parámetro de búsqueda:

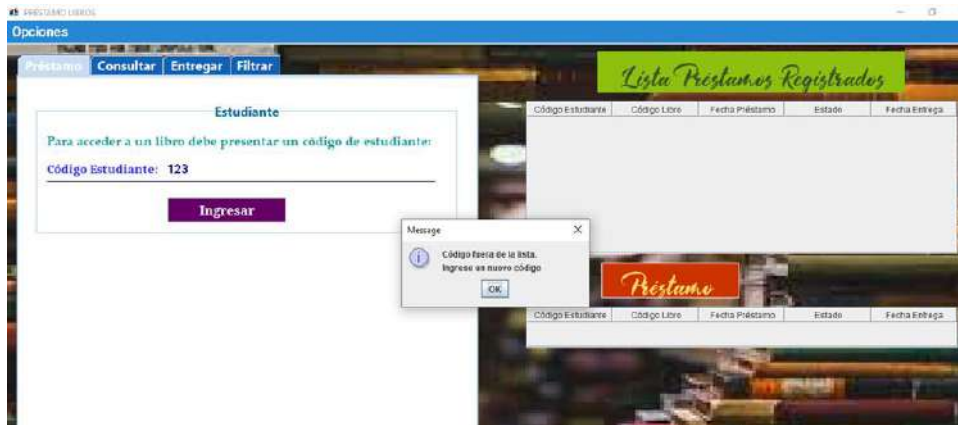


Aparecerán los registros que coincidan con ese atributo.

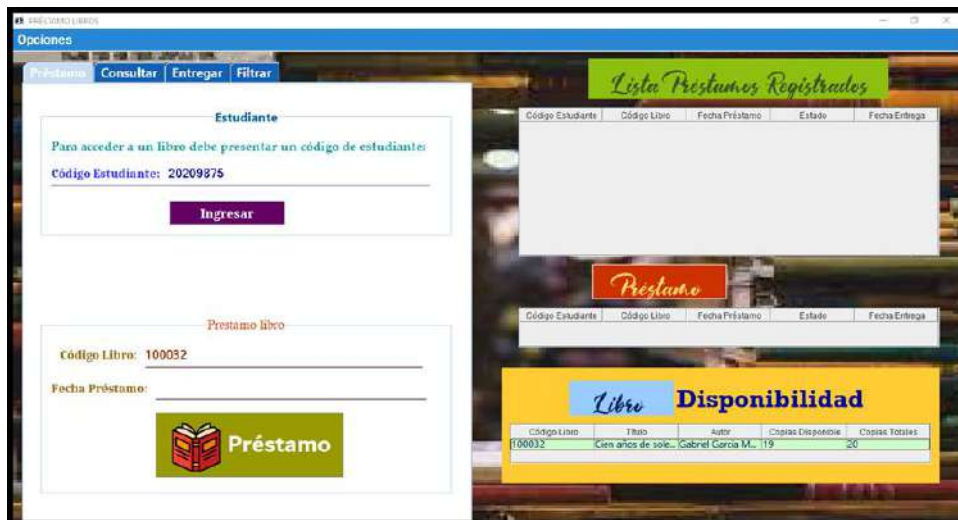
Opción Prestamos (el usuario puede realizar un préstamo, consultar, entregar y filtrar los préstamos) en la parte superior izquierda siempre estará disponible regresar al menú principal y salir, mientras el usuario realice un cambio de panel del jTable los atributos regresan a un estado de “nuevo” (se colocan espacios en blanco) :



Opción Préstamo préstamo, si el código del estudiante no existe no se le puede prestar un libro:



Si ingresa un código del estudiante valido aparece un panel que se debe llenar con información del libro a prestar, si el código del libro existe despliega un apartado con la disponibilidad de copias del mismo.



Para realizar el préstamo, la disponibilidad del libro debe ser de al menos 1, caso contrario no permite el préstamo.



Una vez acepte el registro, se despliega en la lista con estado no entregado.

PRESTAMO LIBROS

Opciones

Préstamo

Consultar

Entregar

Filtrar

Datos entrega

Código Estudiante: 20210071

Código Libro: 100032

Fecha Entrega: 2/12/2023

Entregar

Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
20209875	100032	1/2/2023	No entregado	sin fecha
20210071	100032	27/2/2023	Entregado	2/12/2023

Prestamo

ENTREGADO

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
20210071	100032	27/2/2023	Entregado	2/12/2023

Libro

Disponibilidad

Código Libro	Título	Autor	Copias Disponible	Copias Totales
100032	Cien años de soledad	Gabriel García M...	19	20

El estado del préstamo pasa a entregado, y el número de copias se agrega en 1.

La última pestaña es filtrar incluye un botón que despliega una tabla:

PRESTAMO LIBROS

Opciones

Préstamo

Consultar

Entregar

Filtrar

Permite mostrar Préstamos con características similares:

Filtrar

Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Prestamo

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

PRESTAMO LIBROS

Opciones

Préstamo

Consultar

Entregar

Filtrar

Permite mostrar Préstamos con características similares:

Filtrar

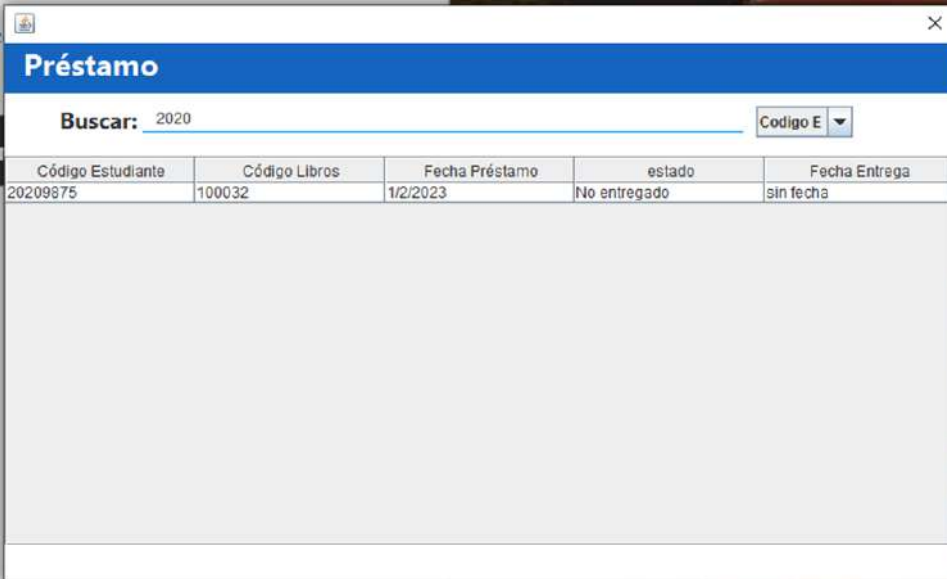
Lista Prestamos Registrados

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

Prestamo

Código Estudiante	Código Libro	Fecha Préstamo	Estado	Fecha Entrega
-------------------	--------------	----------------	--------	---------------

El filtro presenta búsqueda mediante atributos, para activar el textfield se debe elegir un parámetro de búsqueda:



Código Estudiante	Código Libros	Fecha Préstamo	estado	Fecha Entrega
20209875	100032	1/2/2023	No entregado	sin fecha

Aparecerán los registros que coincidan con ese atributo.

CONCLUSIONES Y RECOMENDACIONES

- En este proyecto se utilizó todo recurso previamente analizado en clase y empleamos la capacidad más importante como futuros ingenieros que fue investigar.
- Es de suma importancia verificar, por ejemplo: antes de eliminar que tanto el libro como el estudiante no tengan préstamos, otra verificación es la disponibilidad de libros, o que al ingresar (crear un nuevo estudiante o libro) el código exista o no.
- Trabajar con MySql requiere de estudio para conocer sintaxis y diferentes funciones que permitan la navegación y faciliten el correcto funcionamiento del programa.
- Las tablas que se crean en Java netbeans van almacenando datos, a favor de resolver este inconveniente se elimina cualquier dato antes de ingresar un nuevo, con esto se asegura que la información a mostrar sea la actualizada o ingresada en ese momento.
- Entre las recomendaciones, se puede destacar crear una base de datos mediante phpMyAdmin (es una herramienta de software libre escrita en PHP, destinado a manejar la administración de MySQL), en este caso se opta por usar wampp (es un paquete de software libre, que consiste principalmente en el sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP).
- La cualidad más destacada por quienes desarrollan MySQL es su velocidad y así cómo el software fue diseñado desde un principio, pensando principalmente en la rapidez. - No es caro. MySQL es gratis bajo la licencia GPL de código abierto, y el costo por licencia comercial es muy razonable. - Fácil de usar.
- Las bases de datos nos permiten almacenar, organizar y recuperar grandes cantidades de información de manera eficiente. En este programa se utiliza para almacenar y acceder a los datos de estudiantes y libros de forma estructurada, lo que facilita la administración de la información.
- La nomenclatura que se utilice para cada ingreso de datos debe ser clara y fácil de distinguir pues en el proceso puede presentarse confusión en atributos a enviar para eliminar o cualquier proceso.

- Ser cuidadoso con declaraciones SQL, tanto para la conexión como para las consultas. Una letra minúscula, una tilde lo que sea puede provocar un error, para prevenir revisar el manual de MySQL de la versión que esté utilizando. SI se desea utilizar atributos para realizar búsquedas debe ser consiente si el atributo es un valor entero o un String ese tipo de consideraciones puede ahorrar horas de análisis de errores.
- Para futuros proyecto recomendamos un programa que lleve un historial de registros, es decir hoy alquila una persona, esa persona devuelve el libro ese mismo día, mañana la persona quiere volver a alquilar que se guarde ambos datos tanto el de hoy como el de mañana no solo el actual como lo hace este programa.
- Recomendamos realizar muchas más verificaciones, en los apartados de búsqueda de atributos.

BIBLIOGRAFÍA

Upadhyay, N. (2020, marzo 19). *How to install MySQL database server 8.0.19 on Windows 10*. SQL Shack - Articles about Database Auditing, Server Performance, Data Recovery, and More; SQL Shack.
<https://www.sqlshack.com/how-to-install-mysql-database-server-8-0-19-on-windows-10/>

CS Corner (2022), (<https://www.youtube.com/watch?v=XG0t7ojltyY>)

Oracle (2022) <https://dev.mysql.com/doc/refman/8.0/en/mysql-installer-catalog-dashboard.html#windows-product-dashboard>

ESET

(2022)https://help.eset.com/esmc_install/72/esCL/databaserequirements.html

ESET (2022)https://help.eset.com/esmc_install/72/es-CL/mysql_windows.html

Oracle (2022) <https://dev.mysql.com/downloads/windows/installer/>

Nominalia (2020) <https://www.nominalia.com/asistencia/como-crear-una-base-de-datos-mysql-subir-descargar-windows/>

(N.d.). Www.Uv.Mx. , from <https://www.uv.mx/personal/lizhernandez/files/2013/04/Comandos-mysql.pdf>

Chaparro, E. (2019, January 30). *Programador Novato*. Programador Novato.
<https://www.programadornovato.com/conectar-mysql-con-netbeans/>

(N.d.-b). Mysql.com. from <https://dev.mysql.com/doc/index-connectors.html>

