Proyecto final: Manejo de archivosJorge Peñaloza

Programación Avanzada II

Instituto IACC

18 de octubre de 2021

**Desarrollo**

***Contexto***

Usted ha sido contratado para realizar un sistema bajo ambiente web aplicando sus conocimientos de Programación en Python para la automatización de los procesos de una biblioteca en la ciudad de Temuco. Este sistema deberá permitir al lector conocer la disponibilidad de libros en el sistema para préstamo interno y externo, los días que este libro puede estar fuera de la biblioteca, la cantidad de libros disponibles, los que son para préstamos internos y los que solo son externos, el usuario de la biblioteca deberá visualizar los siguientes campos: nombre del libro, autor, breve descripción del libro, año de publicación, cantidad disponible y colaboración del préstamo. De igual forma debe cumplir con los siguientes requerimientos en su desarrollo:

1. Utilizar como lenguaje de diseño del BackEnd Python para dar solución a situaciones presentadas.
2. Comparar las estructuras de control en situaciones presentadas y seleccionar la mejor opción en beneficio de la solución de problemas.
3. Aplicar arreglos en vectores y matrices para la organización de datos.
4. Promover la aplicación de la programación gráfica con TKinter para la solución del problema presentado.
5. Apoyarse en la utilización de objetos y clases que representen situaciones reales del sistema a realizar.
6. Configurar las operaciones necesarias en la base de datos para interactuar con el lenguaje Python.
7. Aplicar toda la información suministrada en el ramo para solventar el caso planteado.
8. Comprobar la funcionalidad de la aplicación en base al caso desarrollo.

Recuerde documentar su código y cada paso que realice en el documento Word a entregar, también recuerde que debe entregar el diseño de la base de datos, los archivos HTML y los archivos internos del sistema.

***Solución***

A partir de los requerimientos lo primero es general el modelo de base de datos, que se ve implementado en la figura 1.

Este modelo representa 3 tablas independientes llamas *usuarios*, *colaboración* y *libros* que en conjunto permiten gestionar la tabla *préstamo* que contiene la información de cada préstamo realizado, así como también su estado de préstamo.

En código de programación 1 se puede apreciar el archivo *biblioteca.sql* que contiene un respaldo de la base de datos.

Se debe tener en cuenta que el usuario por defecto es:

* Usuario: 1
* Contraseña: 1

Figura 1

*Vista de ejecución de código de programación 1.*

Imagen que contiene Diagrama

Descripción generada automáticamente

Fuente: elaboración propia

Código de programación 1

*Archivo: biblioteca.sql*

-- phpMyAdmin SQL Dump

-- version 5.1.1

-- https://www.phpmyadmin.net/

--

-- Servidor: 127.0.0.1

-- Tiempo de generación: 18-10-2021 a las 03:11:17

-- Versión del servidor: 10.4.21-MariaDB

-- Versión de PHP: 7.4.24

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- Base de datos: `biblioteca`

--

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `colaboracion`

--

CREATE TABLE `colaboracion` (

  `codigo` varchar(20) NOT NULL,

  `nombre` varchar(50) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Volcado de datos para la tabla `colaboracion`

--

INSERT INTO `colaboracion` (`codigo`, `nombre`) VALUES

('1', 'Interno'),

('2', 'IACC'),

('3', 'AIEP'),

('4', 'Duoc');

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `libros`

--

CREATE TABLE `libros` (

  `codigo` varchar(20) NOT NULL,

  `nombre` varchar(50) DEFAULT NULL,

  `autor` varchar(50) DEFAULT NULL,

  `descripcion` varchar(250) DEFAULT NULL,

  `anio` int(4) DEFAULT NULL,

  `cantidad` int(11) DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Volcado de datos para la tabla `libros`

--

INSERT INTO `libros` (`codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad`) VALUES

('A10', 'El capital', 'Karl Marx', 'Crítica de la economía política es un texto teórico fundamental en la filosofía, economía y política de Karl Marx', 1970, 10),

('A20', 'La ética de la libertad', 'Murray Rothbard', 'La ética de la libertad, escrito por el economista e historiador Murray N. Rothbard, publicado por primera vez en 1982, es una exposición lógica y ética sobre la posición política del anarcocapitalismo', 1982, 12);

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `prestamo`

--

CREATE TABLE `prestamo` (

  `id` int(11) NOT NULL,

  `prestamo` int(11) NOT NULL,

  `fecha` date DEFAULT NULL,

  `horario` time DEFAULT NULL,

  `usuario` varchar(50) DEFAULT NULL,

  `codigolibro` varchar(20) DEFAULT NULL,

  `codigocolaboracion` varchar(20) DEFAULT NULL,

  `estado` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Volcado de datos para la tabla `prestamo`

--

INSERT INTO `prestamo` (`id`, `prestamo`, `fecha`, `horario`, `usuario`, `codigolibro`, `codigocolaboracion`, `estado`) VALUES

(3, 1, '2021-10-17', '22:07:18', '2', 'A20', '1: Interno', 'En préstamo'),

(4, 2, '2021-10-17', '22:07:55', '13834570-k', 'A20', '2: IACC', 'En préstamo');

-- --------------------------------------------------------

--

-- Estructura de tabla para la tabla `usuarios`

--

CREATE TABLE `usuarios` (

  `id` int(11) NOT NULL,

  `usuario` varchar(50) NOT NULL,

  `clave` varchar(50) NOT NULL,

  `perfil` char(1) NOT NULL,

  `nombre` varchar(50) NOT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

--

-- Volcado de datos para la tabla `usuarios`

--

INSERT INTO `usuarios` (`id`, `usuario`, `clave`, `perfil`, `nombre`) VALUES

(1, '1', '356a192b7913b04c54574d18c28d46e6395428ab', 'A', 'Jorge Peñaloza'),

(2, '2', 'da4b9237bacccdf19c0760cab7aec4a8359010b0', 'U', 'Ricardo');

--

-- Índices para tablas volcadas

--

--

-- Indices de la tabla `colaboracion`

--

ALTER TABLE `colaboracion`

  ADD PRIMARY KEY (`codigo`);

--

-- Indices de la tabla `libros`

--

ALTER TABLE `libros`

  ADD PRIMARY KEY (`codigo`);

--

-- Indices de la tabla `prestamo`

--

ALTER TABLE `prestamo`

  ADD PRIMARY KEY (`id`),

  ADD KEY `codigobook\_idx` (`codigolibro`),

  ADD KEY `codigocolab\_idx` (`codigocolaboracion`);

--

-- Indices de la tabla `usuarios`

--

ALTER TABLE `usuarios`

  ADD PRIMARY KEY (`id`,`usuario`);

--

-- AUTO\_INCREMENT de las tablas volcadas

--

--

-- AUTO\_INCREMENT de la tabla `prestamo`

--

ALTER TABLE `prestamo`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=5;

--

-- AUTO\_INCREMENT de la tabla `usuarios`

--

ALTER TABLE `usuarios`

  MODIFY `id` int(11) NOT NULL AUTO\_INCREMENT, AUTO\_INCREMENT=3;

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;

Fuente: Elaboración propia

La aplicación está compuesta por 6 scripts en Python que cumplen las siguientes funciones:

* menu.py: Corresponde al programa principal y contiene toda la funcionalidad del entorno gráfico y los procesos de información necesarios para mantener los registros de la aplicación. Este archivo se puede ver implementado en código de programación 2.
* hostBiblioteca.py: Corresponde al script de código de programación 3 que permite leer los datos necesarios para la conexión a la base de datos, estos datos están contenido en un archivo CSV llamado *conexión.dat* cuya información se aprecia en la figura 2 y contiene los siguientes datos:
  + Servidor: 127.0.0.1
  + Usuario: root
  + Contraseña: *No tiene*
  + Base de datos: biblioteca
* usuarios.py: Este script contiene una clase que permite gestionar los datos de la tabla *usuarios* y se puede ver implementada en código de programación 4.
* libros.py: Este script contiene una clase que permite gestionar los datos de la tabla *libros* y se puede ver implementada en código de programación 5.
* colaboradores.py: Este script contiene una clase que permite gestionar los datos de la tabla *colaboracion* y se puede ver implementada en código de programación 6.
* prestamo.py: Este script contiene una clase que permite gestionar los datos de la tabla *prestamo,* y se puede ver implementada en código de programación 7.

Figura 2

*Vista de conexión.dat*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: elaboración propia

Código de programación 2

*Archivo: menu.py*

#from logging import disable

from logging import NullHandler, disable

from tkinter import \*

from tkinter import messagebox as MessageBox

from tkinter import ttk

from usuarios import Usuarios

from libros import Libros

from colaboradores import Colaborador

from datetime import date, datetime

from prestamo import Prestamo

def ayuda():

    MessageBox.showinfo("Ayuda", "Sistema de ayuda esta en desarrollo")

def acercaDe():

    MessageBox.showinfo("Sistema de biblioteca", "Sistema de biblioteca creado por Jorge Peñaloza")

def ventana\_inicio():

    global ventana\_principal

    global filemenu

    global usermenu

    global accesomenu

    global menubar

    pestas\_color="DarkGrey"

    ventana\_principal=Tk()

    ventana\_principal.geometry("600x400") #DIMENSIONES DE LA VENTANA

    ventana\_principal.title("Sistema de biblioteca") #TITULO DE LA VENTANA

    menubar =Menu(ventana\_principal)

    ventana\_principal.config(menu=menubar)

    filemenu = Menu(menubar, tearoff=0)

    filemenu.add\_command(label="Prestamo", command=prestamo)

    filemenu.add\_command(label="Devolucion", command=devolucion)

    filemenu.add\_separator()

    filemenu.add\_command(label="Listado de libros", command=listaLibros )

    filemenu.add\_command(label="Listado de colaboradores", command=listaColaboraciones )

    filemenu.add\_separator()

    filemenu.add\_command(label="Mantenedor de libros", command=crudLibros )

    usermenu = Menu(menubar, tearoff=0)

    usermenu.add\_command(label="Mantenedor de usuarios", command=crudUsuarios)

    usermenu.add\_command(label="Lista de usuarios", command=listaUsuarios)

    accesomenu = Menu(menubar, tearoff=0)

    accesomenu.add\_command(label="Iniciar sesión", command=login)

    accesomenu.add\_command(label="Cerrar sesión", comman=exitLogin)

    accesomenu.entryconfig(index=0, state=NORMAL)

    accesomenu.entryconfig(index=1, state=DISABLED)

    helpmenu = Menu(menubar, tearoff=0)

    helpmenu.add\_command(label="Ayuda", command=ayuda)

    helpmenu.add\_separator()

    helpmenu.add\_command(label="Acerca de...", command=acercaDe)

    menubar.add\_cascade(label="Libros", menu=filemenu)

    menubar.add\_cascade(label="Usuarios", menu=usermenu)

    menubar.add\_cascade(label="Acceso", menu=accesomenu)

    menubar.add\_cascade(label="Ayuda", menu=helpmenu)

    menubar.add\_command(label="Salir", command=ventana\_principal.quit)

    menubar.entryconfig(index=1, state=DISABLED)

    menubar.entryconfig(index=2, state=DISABLED)

    menubar.entryconfig(index=3, state=NORMAL)

    ventana\_principal.mainloop()

def devolucion():

    global entrada\_usuario

    global entrada\_libro

    global libros\_

    global ventana\_devolucion

    ventana\_devolucion = Toplevel(ventana\_principal)

    ventana\_devolucion.title("Devolución de libros")

    ventana\_devolucion.geometry("250x300")

    Label(ventana\_devolucion, text="Introduzca datos de devolucion", bg="LightGreen").pack()

    Label(ventana\_devolucion, text="").pack()

    usuarios\_ = []

    pre = Prestamo()

    usuarios\_prestamo = pre.usuariosConPrestamo()

    for registro in usuarios\_prestamo:

       usuarios\_.append(registro[0])

    etiqueta\_usuario = Label(ventana\_devolucion, text="Usuarios con prestamo pendiente")

    etiqueta\_usuario.pack()

    entrada\_usuario =  ttk.Combobox (ventana\_devolucion, state="readonly")

    entrada\_usuario["values"]   = usuarios\_

    entrada\_usuario.pack()

    Button(ventana\_devolucion, text="Buscar libros de usuario", width=20, height=1, bg="LightGreen", command = librosPrestamo).pack()

    etiqueta\_libro = Label(ventana\_devolucion, text="Libros de usuario con prestamo pendiente")

    etiqueta\_libro.pack()

    entrada\_libro =  ttk.Combobox (ventana\_devolucion, state="readonly")

    entrada\_libro["values"]   = []

    entrada\_libro.pack()

    Button(ventana\_devolucion, text="Efectuar devolucion", width=20, height=1, bg="LightGreen", command = librosDevolucion).pack()

def librosDevolucion():

    usuario\_ = entrada\_usuario.get().strip()

    libro\_ = entrada\_libro.get().strip()

    if usuario\_ == "" or libro\_ == "":

        MessageBox.showerror(message="Debe seleccionar usuario y libro a devolver")

    else:

        libro = Libros()

        (existe, datos) = libro.buscar(libro\_)

        if existe:

            cant = datos[0][5]

            libro.cambiarCantidad(libro\_, int(cant) + 1)

            pres = Prestamo()

            pres.devolver( usuario\_, libro\_)

            MessageBox.showinfo(message="Devolucion exitosa!")

        else:

            MessageBox.showinfo(message="Libro no existe!")

    ventana\_devolucion.destroy()

def librosPrestamo():

    usuario = entrada\_usuario.get()

    libros\_ = []

    pre = Prestamo()

    libros\_prestamo = pre.librosConPrestamo(usuario)

    for registro in libros\_prestamo:

       libros\_.append(registro[0])

    entrada\_libro["values"]   = libros\_

def prestamo():

    global ventana\_prestamo

    global entrada\_fecha

    global fecha\_prestamo

    global entrada\_hora

    global hora\_prestamo

    global usuario\_prestamo

    global tipo\_usuario

    global entrada\_usuario

    global usuario\_prestamo

    global entrada\_prestamo

    global prestamo\_prestamo

    global etiqueta\_colaborador

    global entrada\_colaborador

    global entrada\_libro

    global libro\_prestamo

    global combo

    fecha\_prestamo = StringVar()

    hora\_prestamo = StringVar()

    usuario\_prestamo =StringVar()

    prestamo\_prestamo = IntVar()

    libro\_prestamo = StringVar()

    ventana\_prestamo = Toplevel(ventana\_principal)

    ventana\_prestamo.title("Prestamo de libros")

    ventana\_prestamo.geometry("300x420")

    Label(ventana\_prestamo, text="Introduzca datos de prestamo", bg="LightGreen").pack()

    Label(ventana\_prestamo, text="").pack()

    etiqueta\_prestamo = Label(ventana\_prestamo, text="Prestamo")

    etiqueta\_prestamo.pack()

    entrada\_prestamo = Entry(ventana\_prestamo, textvariable=prestamo\_prestamo)

    entrada\_prestamo.pack()

    entrada\_prestamo.delete(0, END)

    pre = Prestamo()

    (existe, ultimo\_prestamo) = pre.ultimoPrestamo()

    if existe:

        if ultimo\_prestamo[0][0] == None:

            entrada\_prestamo.insert(0, 1)

        else:

            entrada\_prestamo.insert(0, ultimo\_prestamo[0][0])

    else:

        entrada\_prestamo.insert(0, 1)

    entrada\_prestamo.configure(state="readonly")

    etiqueta\_fecha = Label(ventana\_prestamo, text="Fecha")

    etiqueta\_fecha.pack()

    entrada\_fecha = Entry(ventana\_prestamo, textvariable=fecha\_prestamo)

    entrada\_fecha.pack()

    etiqueta\_hora = Label(ventana\_prestamo, text="Hora")

    etiqueta\_hora.pack()

    entrada\_hora = Entry(ventana\_prestamo, textvariable=hora\_prestamo)

    entrada\_hora.pack()

    entrada\_fecha.delete(0, END)

    entrada\_fecha.insert(0,datetime.now().strftime('%Y-%m-%d'))

    entrada\_hora.delete(0, END)

    entrada\_hora.insert(0,datetime.now().strftime('%H:%M:%S'))

    #checkbox\_value = BooleanVar()

    #tipo\_usuario = Checkbutton(ventana\_prestamo, text="Usuario interno ..", variable=checkbox\_value)

    #tipo\_usuario.pack()

    #checkbox\_value.set(TRUE)

    colab = Colaborador()

    datos = colab.listado()

    combo = []

    for reg in datos:

        combo.append(reg[0] + ": " + reg[1])

    etiqueta\_colaborador = Label(ventana\_prestamo, text="Codigo de colaborador")

    etiqueta\_colaborador.pack()

    entrada\_colaborador =  ttk.Combobox (ventana\_prestamo, state="readonly")

    entrada\_colaborador["values"]   = combo

    entrada\_colaborador.pack()

    etiqueta\_usuario = Label(ventana\_prestamo, text="RUT(Externo)/Usuario(Interno) ")

    etiqueta\_usuario.pack()

    entrada\_usuario = Entry(ventana\_prestamo, textvariable=usuario\_prestamo)

    entrada\_usuario.pack()

    etiqueta\_libro = Label(ventana\_prestamo, text="Codigo de libro")

    etiqueta\_libro.pack()

    entrada\_libro = Entry(ventana\_prestamo, textvariable=libro\_prestamo)

    entrada\_libro.pack()

    Label(ventana\_prestamo, text="").pack()

    Button(ventana\_prestamo, text="Efectuar prestamo", width=15, height=1, bg="LightGreen", command = guardarPrestamo).pack()

    Button(ventana\_prestamo, text="Cancelar", width=15, height=1, bg="LightYellow", command = cerrar\_prestamo).pack()

def cerrar\_prestamo():

    ventana\_prestamo.destroy()

def guardarPrestamo():

    prestamo\_ = prestamo\_prestamo.get()

    fecha\_ = fecha\_prestamo.get()

    hora\_ = hora\_prestamo.get()

    usuario\_ = usuario\_prestamo.get()

    codigo\_libro\_ = libro\_prestamo.get()

    codigo\_colaborador = entrada\_colaborador.get()

    if prestamo\_ == "" or fecha\_.strip() == "" or hora\_.strip() == "" or usuario\_.strip() == "" or codigo\_libro\_.strip() == "" or codigo\_colaborador.strip() == "":

        MessageBox.showwarning(message="No pueden haber campos vacios para efectuar prestamo")

    else:

        libro = Libros()

        (existe, datos) = libro.buscar(codigo\_libro\_)

        if existe:

            cant = datos[0][5]

            libro.cambiarCantidad(codigo\_libro\_, int(cant) - 1)

            pres = Prestamo()

            pres.insertar( prestamo\_, fecha\_, hora\_, usuario\_, codigo\_libro\_,codigo\_colaborador,"En préstamo")

            MessageBox.showinfo(message="Prestamo exitoso")

        else:

            MessageBox.showinfo(message="Libro no existe!")

    ventana\_prestamo.destroy()

def crudLibros():

    global ventana\_registro\_libros

    ventana\_registro\_libros = Toplevel(ventana\_principal)

    ventana\_registro\_libros.title("Mantenedor de libros")

    ventana\_registro\_libros.geometry("300x420")

    global codigo\_libro

    global nombre\_libro

    global autor\_libro

    global descripcion\_libro

    global anio\_libro

    global cantidad\_libro

    global entrada\_codigo

    global entrada\_nombreL

    global entrada\_descripcion

    global entrada\_anio

    global entrada\_cantidad

    global entrada\_autor

    codigo\_libro = StringVar()

    nombre\_libro = StringVar()

    autor\_libro = StringVar()

    descripcion\_libro = StringVar()

    anio\_libro = StringVar()

    cantidad\_libro = StringVar()

    Label(ventana\_registro\_libros, text="Introduzca datos", bg="LightGreen").pack()

    Label(ventana\_registro\_libros, text="").pack()

    etiqueta\_codigo = Label(ventana\_registro\_libros, text="Codigo de libro")

    etiqueta\_codigo.pack()

    entrada\_codigo = Entry(ventana\_registro\_libros, textvariable=codigo\_libro)

    entrada\_codigo.pack()

    etiqueta\_nombre = Label(ventana\_registro\_libros, text="Nombre de libro")

    etiqueta\_nombre.pack()

    entrada\_nombreL = Entry(ventana\_registro\_libros, textvariable=nombre\_libro)

    entrada\_nombreL.pack()

    etiqueta\_autor = Label(ventana\_registro\_libros, text="Autor de libro")

    etiqueta\_autor.pack()

    entrada\_autor = Entry(ventana\_registro\_libros, textvariable=autor\_libro)

    entrada\_autor.pack()

    etiqueta\_descripcion = Label(ventana\_registro\_libros, text="Descripcion")

    etiqueta\_descripcion.pack()

    entrada\_descripcion = Entry(ventana\_registro\_libros, textvariable=descripcion\_libro)

    entrada\_descripcion.pack()

    etiqueta\_anio = Label(ventana\_registro\_libros, text="Año")

    etiqueta\_anio.pack()

    entrada\_anio = Entry(ventana\_registro\_libros, textvariable=anio\_libro)

    entrada\_anio.pack()

    etiqueta\_cantidad = Label(ventana\_registro\_libros, text="Cantidad")

    etiqueta\_cantidad.pack()

    entrada\_cantidad = Entry(ventana\_registro\_libros, textvariable=cantidad\_libro)

    entrada\_cantidad.pack()

    Label(ventana\_registro\_libros, text="").pack()

    Button(ventana\_registro\_libros, text="Buscar", width=10, height=1, bg="LightGreen", command = buscar\_libro).pack()

    Button(ventana\_registro\_libros, text="Actualizar", width=10, height=1, bg="LightYellow", command = actualizar\_libro).pack()

    Button(ventana\_registro\_libros, text="Eliminar", width=10, height=1, bg="#F25245", command = eliminar\_libro).pack()

    Button(ventana\_registro\_libros, text="Limpiar", width=10, height=1, bg="Green", command = limpiar\_libro).pack()

def listaColaboraciones():

    global ventana\_lista\_colaborador

    ventana\_lista\_colaborador = Toplevel(ventana\_principal)

    ventana\_lista\_colaborador.title("Lista de colaboradores")

    ventana\_lista\_colaborador.geometry("300x150")

    user = Colaborador()

    datos = user.listado()

    #datos.insert(0,["Nombre de usuario","Usuario","Perfil"])

    #Table(ventana\_lista\_usuarios,datos)

    tabla=ttk.Treeview(ventana\_lista\_colaborador,selectmode=BROWSE,height=10,columns="Nombre" )

    tabla.grid(row=10,column=0,columnspan=2)

    tabla.column("#0",width=50)

    tabla.column("Nombre",width=150)

    tabla.heading("#0",text="Codigo",anchor=CENTER)

    tabla.heading("Nombre",text="Nombre",anchor=CENTER)

    for elemento in datos:

            tabla.insert("",0,text=elemento[0],values=elemento[1])

def buscar\_libro():

    codigo\_ = codigo\_libro.get()

    libro = Libros()

    (existe, datos) = libro.buscar(codigo\_)

    if existe:

        entrada\_codigo.configure (state="readonly")

        entrada\_nombreL.delete(0, END)

        entrada\_nombreL.insert(0,datos[0][1])

        entrada\_autor.delete(0, END)

        entrada\_autor.insert(0,datos[0][2])

        entrada\_descripcion.delete(0, END)

        entrada\_descripcion.insert(0,datos[0][3])

        entrada\_anio.delete(0, END)

        entrada\_anio.insert(0,datos[0][4])

        entrada\_cantidad.delete(0, END)

        entrada\_cantidad.insert(0,datos[0][5])

    else:

        MessageBox.showerror(message="Usuario no existe!")

def actualizar\_libro():

    codigo\_ = codigo\_libro.get()

    nombre\_ = nombre\_libro.get()

    autor\_ = autor\_libro.get()

    descripcion\_ = descripcion\_libro.get()

    anio\_ = anio\_libro.get()

    cantidad\_ = cantidad\_libro.get()

    if codigo\_.strip() == "" or nombre\_.strip() == "" or autor\_.strip() == "" or descripcion\_.strip() == "" or anio\_.strip() == "" or cantidad\_.strip() == "":

        MessageBox.showwarning(message="No pueden haber campos vacios para actualizar")

    else:

        libro = Libros()

        (existe, datos) = libro.buscar(codigo\_)

        if existe:

            libro.actualizar(codigo\_,nombre\_,autor\_,descripcion\_,anio\_,cantidad\_)

            MessageBox.showinfo(message="Libro modificado exitosamente!")

        else:

            libro.insertar(codigo\_,nombre\_,autor\_,descripcion\_,anio\_,cantidad\_)

            MessageBox.showinfo(message="Libro agregado exitosamente!")

        limpiar\_libro()

def eliminar\_libro():

    codigo\_ = codigo\_libro.get()

    nombre\_ = nombre\_libro.get()

    autor\_ = autor\_libro.get()

    mensaje = "Esta seguro de eliminar a "+nombre\_+ " escrito por "+ autor\_

    resp = MessageBox.askquestion(title="Sistema de biblioteca", message=mensaje)

    if resp == "yes":

        libro = Libros()

        libro.eliminar(codigo\_)

    limpiar\_libro()

def limpiar\_libro():

    entrada\_codigo.configure (state=NORMAL)

    entrada\_codigo.delete(0, END)

    entrada\_codigo.insert(0,"")

    entrada\_nombreL.delete(0, END)

    entrada\_nombreL.insert(0,"")

    entrada\_autor.delete(0, END)

    entrada\_autor.insert(0,"")

    entrada\_descripcion.delete(0, END)

    entrada\_descripcion.insert(0,"")

    entrada\_anio.delete(0, END)

    entrada\_anio.insert(0,"")

    entrada\_cantidad.delete(0, END)

    entrada\_cantidad.insert(0,"")

def listaLibros():

    global ventana\_lista\_libros

    ventana\_lista\_libros = Toplevel(ventana\_principal)

    ventana\_lista\_libros.title("Lista de libros")

    ventana\_lista\_libros.geometry("760x250")

    user = Libros()

    datos = user.listado()

    #datos.insert(0,["Nombre de usuario","Usuario","Perfil"])

    #Table(ventana\_lista\_usuarios,datos)

    tabla=ttk.Treeview(ventana\_lista\_libros,selectmode=BROWSE,height=10,columns=("Nombre","Autor","Descripción","Año","Cantidad" ))

    tabla.grid(row=10,column=0,columnspan=2)

    tabla.column("#0",width=50)

    tabla.column("Nombre",width=150)

    tabla.column("Autor",width=100)

    tabla.column("Descripción",width=250)

    tabla.column("Año",width=100)

    tabla.column("Cantidad",width=100)

    tabla.heading("#0",text="Codigo",anchor=CENTER)

    tabla.heading("Nombre",text="Nombre",anchor=CENTER)

    tabla.heading("Autor",text="Autor",anchor=CENTER)

    tabla.heading("Descripción",text="Descripción",anchor=CENTER)

    tabla.heading("Año",text="Año",anchor=CENTER)

    tabla.heading("Cantidad",text="Cantidad",anchor=CENTER)

    for elemento in datos:

            tabla.insert("",0,text=elemento[0],values=(elemento[1], elemento[2], elemento[3], elemento[4], elemento[5]))

def listaUsuarios():

    global ventana\_lista\_usuarios

    ventana\_lista\_usuarios = Toplevel(ventana\_principal)

    ventana\_lista\_usuarios.title("Lista de usuarios")

    ventana\_lista\_usuarios.geometry("402x250")

    user = Usuarios()

    datos = user.listado()

    #datos.insert(0,["Nombre de usuario","Usuario","Perfil"])

    #Table(ventana\_lista\_usuarios,datos)

    tabla=ttk.Treeview(ventana\_lista\_usuarios,selectmode=BROWSE,height=10,columns=("Usuario","Perfil"))

    tabla.grid(row=10,column=0,columnspan=2)

    tabla.column("#0",width=200)

    tabla.column("Usuario",width=100)

    tabla.column("Perfil",width=100)

    tabla.heading("#0",text="Nombre de usuario",anchor=CENTER)

    tabla.heading("Usuario",text="Usuario",anchor=CENTER)

    tabla.heading("Perfil",text="Perfil",anchor=CENTER)

    for elemento in datos:

            tabla.insert("",0,text=elemento[0],values=(elemento[1], elemento[2]))

def crudUsuarios():

    global ventana\_registro

    ventana\_registro = Toplevel(ventana\_principal)

    ventana\_registro.title("Registro")

    ventana\_registro.geometry("300x250")

    global nombre\_usuario

    global nombre\_usuarioC

    global clave

    global entrada\_nombre

    global entrada\_nombreC

    global entrada\_clave

    global perfil

    nombre\_usuario = StringVar()

    nombre\_usuarioC = StringVar()

    clave = StringVar()

    Label(ventana\_registro, text="Introduzca datos", bg="LightGreen").pack()

    Label(ventana\_registro, text="").pack()

    etiqueta\_nombre = Label(ventana\_registro, text="Usuario \* ")

    etiqueta\_nombre.pack()

    entrada\_nombre = Entry(ventana\_registro, textvariable=nombre\_usuario)

    entrada\_nombre.pack()

    etiqueta\_clave = Label(ventana\_registro, text="Contraseña \* ")

    etiqueta\_clave.pack()

    entrada\_clave = Entry(ventana\_registro, textvariable=clave, show='\*')

    entrada\_clave.pack()

    etiqueta\_perfil = Label(ventana\_registro, text="Perfil de usuario")

    etiqueta\_perfil.pack()

    perfil =  ttk.Combobox (ventana\_registro, state="readonly")

    perfil["values"]   = ["A: Administrador","U: Usuario"]

    perfil.pack()

    etiqueta\_nombreC = Label(ventana\_registro, text="Nombre del usuario")

    etiqueta\_nombreC.pack()

    entrada\_nombreC = Entry(ventana\_registro, textvariable=nombre\_usuarioC)

    entrada\_nombreC.pack()

    Label(ventana\_registro, text="").pack()

    Button(ventana\_registro, text="Registrarse", width=10, height=1, bg="LightGreen", command = registro\_usuario).pack()

def exitLogin():

    menubar.entryconfig(index=1, state=DISABLED)

    menubar.entryconfig(index=2, state=DISABLED)

    menubar.entryconfig(index=3, state=NORMAL)

    accesomenu.entryconfig(index=0, state=NORMAL)

    accesomenu.entryconfig(index=1, state=DISABLED)

    filemenu.entryconfig(index=0, state=DISABLED)

    filemenu.entryconfig(index=1, state=DISABLED)

    filemenu.entryconfig(index=3, state=DISABLED)

    filemenu.entryconfig(index=4, state=DISABLED)

    filemenu.entryconfig(index=6, state=DISABLED)

def login():

    global ventana\_login

    ventana\_login = Toplevel(ventana\_principal)

    #ventana\_login = Tk()

    ventana\_login.title("Acceso a la cuenta")

    ventana\_login.geometry("300x250")

    Label(ventana\_login, text="Introduzca nombre de usuario y contraseña").pack()

    Label(ventana\_login, text="").pack()

    global verifica\_usuario

    global verifica\_clave

    verifica\_usuario = StringVar()

    verifica\_clave = StringVar()

    global entrada\_login\_usuario

    global entrada\_login\_clave

    Label(ventana\_login, text="Nombre usuario \* ").pack()

    entrada\_login\_usuario = Entry(ventana\_login, textvariable=verifica\_usuario)

    entrada\_login\_usuario.pack()

    Label(ventana\_login, text="").pack()

    Label(ventana\_login, text="Contraseña \* ").pack()

    entrada\_login\_clave = Entry(ventana\_login, textvariable=verifica\_clave, show= '\*')

    entrada\_login\_clave.pack()

    Label(ventana\_login, text="").pack()

    Button(ventana\_login, text="Acceder", width=10, height=1, command = verifica\_login).pack()

    entrada\_login\_usuario.focus\_set()

def habilitarAdministrador():

    menubar.entryconfig(index=1, state=NORMAL)

    menubar.entryconfig(index=2, state=NORMAL)

    menubar.entryconfig(index=3, state=NORMAL)

    accesomenu.entryconfig(index=0, state=DISABLED)

    accesomenu.entryconfig(index=1, state=NORMAL)

    filemenu.entryconfig(index=0, state=NORMAL)

    filemenu.entryconfig(index=1, state=NORMAL)

    filemenu.entryconfig(index=3, state=NORMAL)

    filemenu.entryconfig(index=4, state=NORMAL)

    filemenu.entryconfig(index=6, state=NORMAL)

def habilitarUsuario():

    menubar.entryconfig(index=1, state=NORMAL)

    menubar.entryconfig(index=2, state=DISABLED)

    menubar.entryconfig(index=3, state=NORMAL)

    accesomenu.entryconfig(index=0, state=DISABLED)

    accesomenu.entryconfig(index=1, state=NORMAL)

    filemenu.entryconfig(index=0, state=DISABLED)

    filemenu.entryconfig(index=1, state=DISABLED)

    filemenu.entryconfig(index=3, state=NORMAL)

    filemenu.entryconfig(index=4, state=NORMAL)

    filemenu.entryconfig(index=6, state=DISABLED)

def verifica\_login():

    usuario1 = verifica\_usuario.get()

    clave1 = verifica\_clave.get()

    acceso = Usuarios()

    (existencia, datos) = acceso.buscar(usuario1,clave1)

    entrada\_login\_usuario.delete(0, END)

    entrada\_login\_clave.delete(0, END)

    if existencia:

        titulo = "Sistema de biblioteca con acceso a "

        titulo += datos[0][0] + "[[ perfil: "+ datos[0][1] +"]]"

        if datos[0][1] == "A":

            habilitarAdministrador()

        if datos[0][1] == "U":

            habilitarUsuario()

        ventana\_principal.title(titulo)

        ventana\_login.destroy()

    else:

        MessageBox.showinfo(message="Usuario no existe o hay error de contraseña")

        ventana\_login.destroy()

def registro\_usuario():

    usuario\_info = entrada\_nombre.get()

    nombre\_info = entrada\_nombreC.get()

    clave\_info = entrada\_clave.get()

    perilUsuario = perfil.get()

    acceso = Usuarios()

    (existencia, datos) = acceso.buscarUsuario(usuario\_info)

    if existencia:

        mensaje = "Usuario " + datos[0][0] + " ya existe "

        mensaje += "¿Ésta seguro de cambiar contraseña ?"

        resp = MessageBox.askquestion(title="Sistema de biblioteca", message=mensaje)

        if resp == "yes":

            try:

                acceso.actualizar(usuario\_info,clave\_info,perilUsuario[0:1],nombre\_info)

            except:

                MessageBox.showerror(message="Error al ingresar Usuario")

    else:

        mensaje = "Usuario " + nombre\_info + " NO existe "

        mensaje += "¿Ésta seguro de agregar usuario ?"

        resp = MessageBox.askquestion(title="Sistema de biblioteca", message=mensaje)

        if resp == "yes":

            try:

                acceso.insertar(usuario\_info,clave\_info,perilUsuario[0:1],nombre\_info)

            except:

                MessageBox.showerror(message="Error al ingresar Usuario")

    entrada\_nombre.delete(0, END)

    entrada\_clave.delete(0, END)

    ventana\_registro.destroy()

ventana\_inicio()  #EJECUCIÓN DE LA VENTANA DE INICIO.

Fuente: Elaboración propia

Código de programación 3

*Archivo: hostBiblioteca.py*

import csv

def traerHost():

    file = open('conexion.dat', "r")

    reader = csv.reader(file, delimiter=',', quotechar=',', quoting=csv.QUOTE\_MINIMAL)

    sw = False

    for row in reader:

        if reader.line\_num == 1:

            return tuple(row)

        elif reader.line\_num > 1:

            return False

Fuente: Elaboración propia

Código de programación 4

*Archivo: usuarios.py*

#import os

import hostBiblioteca as hb

import mysql.connector

import hashlib

class Usuarios:

    def \_\_init\_\_(self):

        conexion = hb.traerHost()

        self.servidor = conexion[0]

        self.usuario = conexion[1]

        self.clave = conexion[2].strip()

        self.db = conexion[3]

    def buscar(self, usuario, clave):

        claveEnBytes = str.encode(clave)

        claveCodificada = hashlib.new("sha1", claveEnBytes)

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT nombre, perfil  FROM `usuarios` "

        query += "WHERE usuario  = '"+usuario+"' AND "

        query += "clave = '" + claveCodificada.hexdigest() +"'"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        filas = mycursor.rowcount

        if( filas == 1):

            return (True, datos)

        else:

            return (False, datos)

    def buscarUsuario(self, usuario):

        #claveEnBytes = str.encode(clave)

        #claveCodificada = hashlib.new("sha1", claveEnBytes)

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT nombre, perfil  FROM `usuarios` "

        query += "WHERE usuario  = '"+usuario+"'"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        filas = mycursor.rowcount

        if( filas == 1):

            return (True, datos)

        else:

            return (False, datos)

    def insertar(self, usuario, clave, perfil, nombre):

        claveEnBytes = str.encode(clave)

        claveCodificada = hashlib.new("sha1", claveEnBytes)

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "INSERT INTO `usuarios` (`id`,`usuario`, `clave`, `perfil`, `nombre`)"

        query += " VALUES (NULL,'"+usuario+"', '"+claveCodificada.hexdigest()

        query += "', '"+perfil+"','"+nombre+"');"

        #print(query)

        mycursor.execute(query)

        mydb.commit()

    def actualizar(self, usuario, clave, perfil, nombre):

        claveEnBytes = str.encode(clave)

        claveCodificada = hashlib.new("sha1", claveEnBytes)

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "UPDATE `usuarios` SET `clave` = '"+claveCodificada.hexdigest()

        query += "', `perfil` = '"+perfil

        query += "', `nombre` = '"+nombre+"' WHERE `usuarios`.`usuario` = '"+usuario+"';"

        #print(query)

        mycursor.execute(query)

        mydb.commit()

    def listado(self):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `nombre`,`usuario`, IF(`perfil` = 'A','Administrador',IF(`perfil` = 'U','Usuario','ERROR')) AS perfil FROM `usuarios` WHERE 1;"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        return datos

Fuente: Elaboración propia

Código de programación 5

*Archivo: libros.py*

#import os

import hostBiblioteca as hb

import mysql.connector

class Libros:

    def \_\_init\_\_(self):

        conexion = hb.traerHost()

        self.servidor = conexion[0]

        self.usuario = conexion[1]

        self.clave = conexion[2].strip()

        self.db = conexion[3]

    def buscar(self, codigo):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad` FROM `libros`"

        query += " WHERE codigo = '"+codigo+"';"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        filas = mycursor.rowcount

        if( filas == 1):

            return (True, datos)

        else:

            return (False, datos)

    def eliminar(self, codigo):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "DELETE FROM `libros` WHERE `codigo` = \'"+codigo+"\'"

        mycursor.execute(query)

        mydb.commit()

    def insertar(self, codigo, nombre, autor, descripcion, anio,cantidad):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "INSERT INTO `libros` (`codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad`) "

        query += " VALUES ('"+codigo+"',"

        query += " '"+nombre+"', "

        query += "'"+autor+"', "

        query += "'"+descripcion+"', "

        query += "'"+anio+"', "

        query += "'"+cantidad+"')"

        #INSERT INTO `libros` (`codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad`) VALUES ('2', '2', '2', '2', '2', '2');

        mycursor.execute(query)

        mydb.commit()

    def actualizar(self, codigo, nombre, autor, descripcion, anio,cantidad):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "UPDATE `libros` SET `nombre` = '"+nombre

        query += "', `autor` = '"+autor

        query += "', `descripcion` = '"+descripcion

        query += "', `anio` = '"+anio

        query += "', `cantidad` = '"+cantidad

        query += "' WHERE `codigo` = '"+codigo+"';"

        #UPDATE `libros` SET `nombre` = 'J', `autor` = 'Richard Bach1', `descripcion` = 'Juan', `anio` = '1971', `cantidad` = '10' WHERE `libros`.`codigo` = 'A10';

        #print(query)

        mycursor.execute(query)

        mydb.commit()

    def cambiarCantidad(self,codigo, cantidad):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "UPDATE `libros` SET `cantidad` = '"+str(cantidad)

        query += "' WHERE `codigo` = '"+codigo+"';"

        #UPDATE `libros` SET `nombre` = 'J', `autor` = 'Richard Bach1', `descripcion` = 'Juan', `anio` = '1971', `cantidad` = '10' WHERE `libros`.`codigo` = 'A10';

        #print(query)

        mycursor.execute(query)

        mydb.commit()

    def listado(self):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad` FROM `libros` WHERE 1;"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        return datos

Fuente: Elaboración propia

Código de programación 6

*Archivo: colaboradores.py*

#import os

import hostBiblioteca as hb

import mysql.connector

class Colaborador:

    def \_\_init\_\_(self):

        conexion = hb.traerHost()

        self.servidor = conexion[0]

        self.usuario = conexion[1]

        self.clave = conexion[2].strip()

        self.db = conexion[3]

    def listado(self):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `codigo`, `nombre` FROM `colaboracion` WHERE 1 ORDER BY codigo DESC;"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        return datos

Fuente: Elaboración propia

Código de programación 7

*Archivo: prestamo.py*

import hostBiblioteca as hb

import mysql.connector

class Prestamo:

    def \_\_init\_\_(self):

        conexion = hb.traerHost()

        self.servidor = conexion[0]

        self.usuario = conexion[1]

        self.clave = conexion[2].strip()

        self.db = conexion[3]

    def ultimoPrestamo(self):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT MAX(prestamo)+1 AS registro   FROM `prestamo`;"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        filas = mycursor.rowcount

        if( filas == 1):

            return (True, datos)

        else:

            return (False, datos)

    def buscar(self, codigo):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad` FROM `libros`"

        query += " WHERE codigo = '"+codigo+"';"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        filas = mycursor.rowcount

        if( filas == 1):

            return (True, datos)

        else:

            return (False, datos)

    def eliminar(self, codigo):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "DELETE FROM `libros` WHERE `codigo` = \'"+codigo+"\'"

        mycursor.execute(query)

        mydb.commit()

    def insertar(self, prestamo, fecha, horario, usuario, codigolibro,codigocolaboracion,estado):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        #INSERT INTO `prestamo` (`id`, `prestamo`, `fecha`, `horario`, `usuario`, `codigolibro`, `codigocolaboracion`, `estado`) VALUES (NULL, '1', '2021-10-13', '20:30:53', 'jorge', 'A10', '1', 'En préstamo');

        query = "INSERT INTO `prestamo` (`id`, `prestamo`, `fecha`, `horario`, `usuario`, `codigolibro`, `codigocolaboracion`, `estado`) "

        query += " VALUES ( NULL, "

        query += "'"+str(prestamo)+"',"

        query += " '"+fecha+"', "

        query += "'"+horario+"', "

        query += "'"+usuario+"', "

        query += "'"+codigolibro+"', "

        query += "'"+codigocolaboracion+"', "

        query += "'"+estado+"')"

        #INSERT INTO `libros` (`codigo`, `nombre`, `autor`, `descripcion`, `anio`, `cantidad`) VALUES ('2', '2', '2', '2', '2', '2');

        mycursor.execute(query)

        mydb.commit()

    def devolver(self, usuario, codigolibro ):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "UPDATE `prestamo` SET `estado` = 'Devuelto' "

        query += "WHERE `usuario` = '"+usuario+"' AND `codigolibro` = '"+codigolibro+"';"

        mycursor.execute(query)

        mydb.commit()

    def usuariosConPrestamo(self):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `usuario` FROM `prestamo` WHERE estado = 'En préstamo' GROUP BY usuario;"

        mycursor.execute(query)

        #print(query)

        datos = mycursor.fetchall()

        return datos

    def librosConPrestamo(self, usuario):

        mydb = mysql.connector.connect( host=self.servidor, user=self.usuario, password=self.clave,database=self.db )

        mycursor = mydb.cursor()

        mycursor.execute("USE "+self.db+";")

        query = "SELECT `codigolibro` FROM `prestamo` WHERE "

        query += "estado = 'En préstamo' AND usuario ='"+usuario+"' "

        query += "GROUP BY codigolibro;"

        mycursor.execute(query)

        datos = mycursor.fetchall()

        return datos

Fuente: Elaboración propia

***Funcionalidad De Aplicación***

La aplicación comienza con la vista de la figura 3 en la cual se tiene un menú horizontal de acceso para iniciar la sesión, para su primer inicio utilice usuario 1 y contraseña 1, para luego gestionar sus propios usuarios.

Figura 3

*Vista inicial de aplicación de biblioteca*

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Fuente: Elaboración propia

En la figura 4 se puede ver el login para acceder a la aplicación.

Figura 4

*Login para acceder a la aplicación*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Si se desea modificar o crear un usuario, puede realizarlo en mantenedor de usuarios de la figura 5, en donde se creará un administrador con nombre y contraseña *iacc* tan cómo ve en la figura 6*.*

Figura 5

*Login para acceder a la aplicación*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

Fuente: Elaboración propia

Figura 6

*Mantenedor de usuario, permite creación y modificación.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

En la figura 7 se puede como se crea un usuario llamado alumno con perfil de usuario.

Figura 6

*Mantenedor de usuario, permite creación y modificación.*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Según el tipo de usuario podrá ver ciertas opciones de menú en la aplicación, por ejemplo, el administrador iacc tiene acceso a todas las opciones del sistema tal como se ve en las figura 7, 8 y 9. Las opciones de acceso, ayuda y salir están permitidas para todos los usuarios.

Figura 7

*Vista de usuario IACC*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 8

*Vista de usuario IACC*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Fuente: Elaboración propia

Figura 9

*Vista de usuario IACC*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Si se ingresa con el usuario *alumno* que tiene solo acceso de usuario básico se puede ver el acceso a las opciones *listado de libros* y *listado de colaboradores* del sistema tal como se ve en la figura 10. Las opciones de acceso, ayuda y salir están permitidas para todos los usuarios.

Figura 10

*Vista de usuario alumno genérico*

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Fuente: Elaboración propia

Las ventanas de Listado de libros y Listado de colaboradores de la figura 10 tienen acceso de ver los datos de las tablas sin restricciones ya que son de uso público.

En la figura 11 se puede ver los colaboradores existentes, además de tener incluido en la base de datos al usuario interno.

Figura 11

*Vista de Listado de colaboradores*

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

Fuente: Elaboración propia

En la figura 12 se puede ver el listado de libros existentes justo a sus respectivos stock.

Figura 12

*Vista de Listado de colaboradores*

Texto

Descripción generada automáticamente

Fuente: Elaboración propia

Al ingresar con el usuario *iacc* que esta con estado de *administrador* se puede acceder además a *Préstamo*, *Devolución* y *Mantenedor de libros* de la figura 13.

Figura 13

*Vista de Préstamo, Devolución y Mantenedor de libros*

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

Fuente: Elaboración propia

Al ingresar a *Mantenedor de libros de la figura 14-A*, se puede apreciar que el mantenedor tiene 4 opciones:

* Buscar: Esta opción permite buscar un libro cuyo código ya se conozca. Cuando el libro existe, la entrada de datos de código de libro cambia su estado a solo lectura para evitar que modifique el código del libro por error. Se puede apreciar al estado de *solo lectura* en la figura 14-B
* Actualizar: Esta opción permite actualizar o insertar un libro, cuando el código de libro es nuevo, se agregará el libro a la base de datos y cuando el código del libro no exista, se actualizará la información según los datos modificados por el usuario.
* Eliminar: Permite eliminar un libro, pero antes de esto debe ingresar el código para poder eliminarlo de la base de datos.
* Limpiar: Limpia los campos de todas las entradas de texto y cambia el estatus de código de libro a *lectura/escritura*

Figura 14

*Vista de Mantenedor de libros en formas A y B respectivamente*

*A B*

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Cuando se desea crear un préstamo se debe tener en cuenta que la opción de usuario interno solo se debe usar para personal o alumnos de la propia institución, solo se debe realizar prestamos externo a alumno que sean de la instituciones definidas en colaboradores tal como se ve en la figura 15.

Al efectuar el préstamo se puede ver en la figura 16 como se visualiza un préstamo exitoso, además en la figura 17 se puede la tabla cuando el prestamos fue ingresado, se debe considerar que cada que se realiza un préstamo el stock de libros baja.

Cuando se realiza una devolución de libros siempre saldrán solo usuarios, tal como se ve en la figura 18, que deben algún libro en la lista desplegable de usuarios, luego de seleccionar el usuario que devolverá libros, debe presionar Buscar libros de usuario para que se actualice la lista de libros en préstamo de ese usuario.

Figura 15

*Vista de Prestamos*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 16

*Vista de Préstamo efectuado*

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 17

*Vista de tabla préstamo después de efectuar un prestamos exitoso*

Interfaz de usuario gráfica

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 18

*Vista de devolución de libros*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Ya seleccionado el usuario y posteriormente el libro, puede efectuar la devolución tal se ve en la figura 19. Cuando la devolución se realiza ocurren 2 eventos se actualiza la tabla *prestamo* cambiando el estado del préstamo a Devuelto y cambia el stock de libros.

El stock original de libro con código A20 es 13, pero en la base existe aun un libro en préstamo y por eso es que el stock de la figura 21 muestra que solamente tiene 13 copias de ese libro en stock.

Figura 19

*Vista de devolución de libros*

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 20

*Tabla de prestamos*

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Figura 21

*Lista de libros*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Finalmente se debe recordar que existe una lista de usuarios vista en la figura 22 y en la figura 23 se puede ver la lista de usuarios con sus contraseñas encriptadas.

Figura 22

*Lista de usuarios*

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Fuente: Elaboración propia

Figura 23 en phpMyAdmin

*Lista de usuarios*

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Fuente: Elaboración propia

Bibliografía

IACC. (2021). EVALUACIÓN FINAL.