

EC0835

Ejecución de software con codificación de comandos y datos orientada a objetos

Manual Técnico

Desarrollado por:
Alejandro Tello Hernandez

CESUN Universidad
2026

Manual Técnico

Sistema Académico EC0835

Alejandro Tello Hernández

14 de junio de 2026

Índice

1. Introducción 2

2. Objetivo general del sistema 2

3. Objetivos específicos del sistema 2

4. Requerimientos de instalación 2

5. Diagrama de clases 3

6. Diagrama entidad-relación 3

7. Desarrollo (explicación técnica del funcionamiento) 4

8. Código de la aplicación 5

1. Introducción

El Sistema Académico EC0835 es una aplicación de escritorio desarrollada para gestionar estudiantes, cursos e inscripciones. El software está diseñado bajo una arquitectura monolítica, utilizando Python puro para la lógica e interfaz gráfica (GUI), y SQLite para la persistencia de datos local, asegurando la autonomía de la aplicación en entornos aislados.

2. Objetivo general del sistema

Proveer una herramienta informática autónoma e intuitiva para la gestión académica, permitiendo llevar el control eficiente de los estudiantes, el catálogo de cursos ofertados y el registro de inscripciones, asegurando la persistencia y la integridad de la información mediante una base de datos local.

3. Objetivos específicos del sistema

- Desarrollar una interfaz gráfica de usuario (GUI) amigable dividida modularmente mediante pestañas.
- Implementar un sistema de base de datos relacional (SQLite) para el almacenamiento local sin requerir instalación de servidores.
- Realizar operaciones de creación, lectura, actualización y eliminación lógica (CRUD completo) de las entidades del sistema.
- Validar la integridad de los datos evitando registros duplicados en llaves únicas (e.g., correos electrónicos).
- Generar reportes tabulares de inscripciones activas con capacidad de exportación a formato CSV.
- Asegurar la trazabilidad del sistema y de los datos mediante la generación de logs automáticos y respaldos de la base de datos por cada modificación.

4. Requerimientos de instalación

Entorno de Desarrollo y Ejecución:

- **Sistema Operativo:** Windows 10 o superior (para la ejecución del binario compilado).

- **Lenguaje Base:** Python 3.8+ (en caso de ejecutarse directamente desde el código fuente).
- **Dependencias y Librerías Nativas (incluidas por defecto):**
  - `tkinter` (Interfaz Gráfica)
  - `sqlite3` (Base de Datos)
  - `logging`, `csv`, `os`, `shutil`, `datetime` (Auditoría, importación/exportación y utilidades)
- **Dependencias Externas:**
  - `pyinstaller` (requerida exclusivamente para la etapa de compilación del ejecutable `.exe`).

5. Diagrama de clases

La aplicación opera bajo un enfoque modular y funcional estructurado. El siguiente diagrama representa la interacción entre las Vistas Gráficas (UI), el controlador de base de datos (`SistemaSQLite`) y los módulos de utilería (`AuditoriaLogs` y `GestorBackups`), demostrando una fuerte cohesión de funciones.

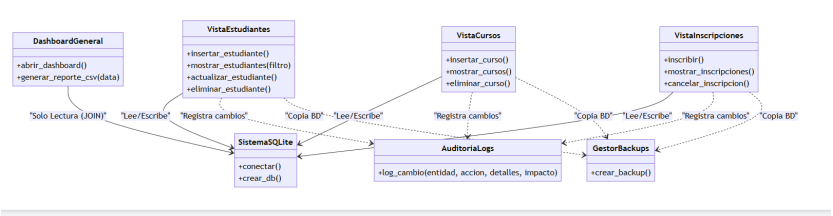

Figura 1: Diagrama de Clases (Módulos)

6. Diagrama entidad-relación

La base de datos relacional consta de tres tablas principales que implementan integridad referencial:

- `estudiantes`: Llave primaria autoincrementable.
- `cursos`: Llave primaria autoincrementable.

- `inscripciones`: Entidad asociativa (puente) que resuelve la cardinalidad de muchos a muchos mediante el uso de llaves foráneas apuntando a estudiantes y cursos.

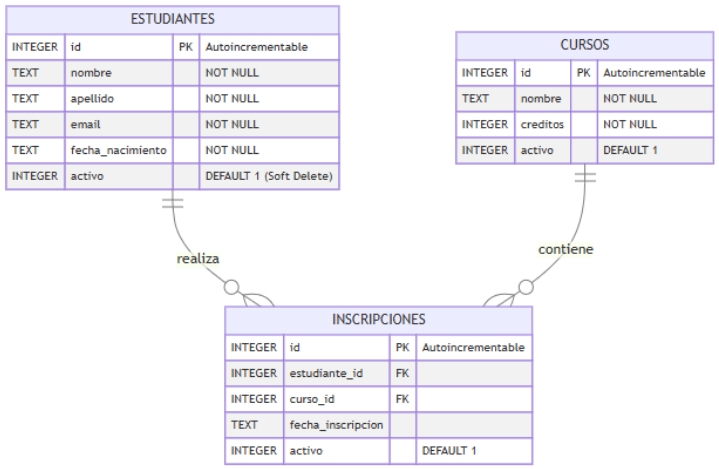

Figura 2: Diagrama Entidad-Relación

7. Desarrollo (explicación técnica del funcionamiento)

El funcionamiento del sistema se rige bajo los siguientes principios y decisiones técnicas:

1. **Persistencia y Control de Errores (SQLite):** El módulo de conexión valida la existencia de `academia.db`. Si no existe, genera el DDL (Data Definition Language) para crear la estructura e inserta datos semillas automatizados (`estudiantes_data`). Toda instrucción DML (INSERT, UPDATE) utiliza variables de sustitución paramétricas (`?`), lo cual previene vulnerabilidades de Inyección SQL.

2. **Interfaz de Usuario (Tkinter):** La interfaz hace uso intensivo del componente `ttk.Notebook` para distribuir el estado de la aplicación sin recargar ventanas, y `ttk.Treeview` para renderizar colecciones de datos. Los eventos de usuario están vinculados mediante eventos (`«TreeviewSelect»`), conectando el flujo de interacción gráfica con los controladores internos.

3. **Manejo de Estados (Soft Delete):** A nivel arquitectónico, la operación lógica de eliminación no borra la tupla permanentemente (`DELETE FROM`). A su vez, muta una bandera de estado booleana (`activo = 0`). Las lecturas subsecuentes están enmascaradas con cláusulas `WHERE activo = 1`. Esto resguarda los registros para auditoría en proyectos escalables.

4. **Auditoría Transaccional e Histórica:** El sistema suscribe eventos ante la confirmación exitosa de una transacción modificadora (`conn.commit()`). En cascada, invoca funciones de `logging` en un fichero plano `logs/sistema.log` y paralelamente invoca la librería `shutil` clonando la imagen de la base de datos hacia el directorio `backups/`.

5. **Compilación Externa:** El despliegue a producción requiere la compilación binaria con la biblioteca externa `PyInstaller` invocando los parámetros `-onefile` (generación singular de artefacto) y `-windowed` (supresión del emulador de terminal).

8. Código de la aplicación

A continuación, se adjunta el código fuente íntegro correspondiente al programa principal (`main.py`):

```
1
2  # ============================================================
3  # SISTEMA ACAD MICO EC0835
4  # Alejandro Tello Hern ndez
5  # CESUN Universidad
6  # ============================================================
7
8  import tkinter as tk
9  from tkinter import ttk, messagebox
10 import sqlite3
11 import logging
12 import os
13 import shutil
14 import csv
15 from tkinter import filedialog
16 from datetime import datetime
17
18 # ============================================================
19 # CONFIG
20 # ============================================================
21
22 DB_NAME = "academia.db"
```

```
23
24 # ============================================================
25 # LOGS
26 # ============================================================
27
28 if not os.path.exists("logs"):
29     os.makedirs("logs")
30
31 logging.basicConfig(
32     filename="logs/sistema.log",
33     level=logging.INFO,
34     format="%(asctime)s - %(levelname)s - %(message)s"
35 )
36
37 def log_cambio(entidad, accion, detalles, implicaciones):
38     mensaje = f"Entidad: {entidad} | Accion: {accion} |
    Detalles: {detalles} | Impacto en BD: {implicaciones}"
39     logging.info(mensaje)
40
41 # ============================================================
42 # BACKUPS
43 # ============================================================
44
45 if not os.path.exists("backups"):
46     os.makedirs("backups")
47
48 # ============================================================
49 # CONEXION
50 # ============================================================
51
52 def conectar():
53     return sqlite3.connect(DB_NAME)
54
55 # ============================================================
56 # CREAR DB
57 # ============================================================
58
59 def crear_db():
60
61     db_existe = os.path.exists(DB_NAME)
62
63     conn = conectar()
64
65     cursor = conn.cursor()
66
67     cursor.execute("""
68     CREATE TABLE IF NOT EXISTS estudiantes(
69         id INTEGER PRIMARY KEY AUTOINCREMENT,
70         nombre TEXT NOT NULL,
```

```
        apellido TEXT NOT NULL,
        email TEXT NOT NULL,
        fecha_nacimiento TEXT NOT NULL,
        activo INTEGER DEFAULT 1
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS cursos(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        nombre TEXT NOT NULL,
        creditos INTEGER NOT NULL,
        activo INTEGER DEFAULT 1
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS inscripciones(
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        estudiante_id INTEGER,
        curso_id INTEGER,
        fecha_inscripcion TEXT,
        activo INTEGER DEFAULT 1,
        FOREIGN KEY(estudiante_id) REFERENCES estudiantes(id),
        FOREIGN KEY(curso_id) REFERENCES cursos(id)
    )
    """)

    if not db_existe:
        estudiantes_data = [
            ("Juan", "Perez", "juan@test.com", "2000-01-01"),
            ("Maria", "Gomez", "maria@test.com", "1999-05-12"),
            ("Carlos", "Ruiz", "carlos@test.com", "2001-08-23"),
            ("Ana", "Lopez", "ana@test.com", "2002-11-30"),
            ("Luis", "Torres", "luis@test.com", "1998-03-15")
        ]
        cursor.executemany("INSERT INTO estudiantes (nombre, apellido, email, fecha_nacimiento) VALUES (?, ?, ?, ?)", estudiantes_data)

        cursos_data = [
            ("Matematicas", 5),
            ("Historia", 4),
            ("Programacion", 6),
            ("Fisica", 5),
            ("Quimica", 4)
```

```
        ]
        cursor.executemany("INSERT INTO cursos (nombre, creditos) VALUES (?, ?)", cursos_data)

        inscripciones_data = [
            (1, 1, "2023-01-10"),
            (2, 2, "2023-01-11"),
            (3, 3, "2023-01-12"),
            (4, 4, "2023-01-13"),
            (5, 5, "2023-01-14")
        ]
        cursor.executemany("INSERT INTO inscripciones (estudiante_id, curso_id, fecha_inscripcion) VALUES (?, ?, ?)", inscripciones_data)

    conn.commit()
    conn.close()

crear_db()

# ============================================================
# BACKUP
# ============================================================

def crear_backup():

    try:

        fecha = datetime.now().strftime("%Y%m%d_%H%M%S")

        destino = f"backups/academia_{fecha}.db"

        shutil.copy(DB_NAME, destino)

        logging.info("Backup creado")

    except Exception as e:

        logging.error(str(e))

# ============================================================
# VENTANA
# ============================================================

ventana = tk.Tk()

ventana.title("Sistema Acad mico EC0835")

ventana.geometry("1450x850")
```

```
161
162 ventana.configure(bg="#1f2937")
163
164 # ============================================================
165 # ESTILOS
166 # ============================================================
167
168 style = ttk.Style()
169
170 style.theme_use("clam")
171
172 style.configure(
173     "Treeview",
174     background="#ffffff",
175     foreground="black",
176     rowheight=28,
177     fieldbackground="#ffffff",
178     font=("Segoe UI", 10)
179 )
180
181 style.configure(
182     "Treeview.Heading",
183     font=("Segoe UI", 10, "bold")
184 )
185
186 # ============================================================
187 # NOTEBOOK
188 # ============================================================
189
190 tabs = ttk.Notebook(ventana)
191
192 tabs.pack(fill="both", expand=True, padx=10, pady=10)
193
194 tab_estudiantes = tk.Frame(tabs, bg="#f3f4f6")
195 tab_cursos = tk.Frame(tabs, bg="#f3f4f6")
196 tab_inscripciones = tk.Frame(tabs, bg="#f3f4f6")
197
198 tabs.add(tab_estudiantes, text="Estudiantes")
199 tabs.add(tab_cursos, text="Cursos")
200 tabs.add(tab_inscripciones, text="Inscripciones")
201
202 # ============================================================
203 # FUNCIONES ESTUDIANTES
204 # ============================================================
205
206 def limpiar_estudiante():
207
208     entry_id.config(state="normal")
209     entry_id.delete(0, tk.END)
```

```
    entry_id.config(state="readonly")

    entry_nombre.delete(0, tk.END)
    entry_apellido.delete(0, tk.END)
    entry_email.delete(0, tk.END)

    combo_dia.set("1")
    combo_mes.set("1")
    combo_anio.set("2000")

def mostrar_estudiantes(filtro=""):

    tree_estudiantes.delete(*tree_estudiantes.get_children())

    conn = conectar()

    cursor = conn.cursor()

    if filtro == "":

        cursor.execute("""
        SELECT id,nombre,apellido,email,fecha_nacimiento
        FROM estudiantes
        WHERE activo = 1
        ORDER BY nombre
        """)

    else:

        cursor.execute("""
        SELECT id,nombre,apellido,email,fecha_nacimiento
        FROM estudiantes
        WHERE activo = 1
        AND nombre LIKE ?
        ORDER BY nombre
        """, (
            f"%{filtro}%",
        ))

    contador = 1

    for row in cursor.fetchall():

        tree_estudiantes.insert(
            "",
            "end",
            values=(
                contador,
                row[1],
```

```
                row[2],
                row[3],
                row[4]
            ),
            tags=(row[0],)
        )

        contador += 1

    conn.close()

def insertar_estudiante():

    nombre = entry_nombre.get().strip()
    apellido = entry_apellido.get().strip()
    email = entry_email.get().strip()

    fecha = f"{combo_anio.get()}-{combo_mes.get()}-{combo_dia.get()}"

    if nombre == "" or apellido == "" or email == "":

        messagebox.showwarning(
            "Validaci n",
            "Todos los campos son obligatorios"
        )

        return

    conn = conectar()

    cursor = conn.cursor()

    cursor.execute("""
    SELECT *
    FROM estudiantes
    WHERE email = ?
    AND activo = 1
    """, (
        email,
    ))

    existe = cursor.fetchone()

    if existe:

        messagebox.showwarning(
            "Validaci n",
            "Ya existe un estudiante activo con ese correo"
```

```
307         )
308 
309         conn.close()
310 
311         return
312 
313     try:
314 
315         cursor.execute("""
316         INSERT INTO estudiantes(
317             nombre,
318             apellido,
319             email,
320             fecha_nacimiento
321         )
322         VALUES(?,?,?,?)
323         """, (
324             nombre,
325             apellido,
326             email,
327             fecha
328         ))
329 
330         conn.commit()
331         crear_backup()
332         log_cambio(
333             "Estudiante",
334             "INSERT",
335             f"Se agrego estudiante: {nombre} {apellido} ({
    email})",
336             "El estudiante ahora puede ser inscrito a cursos
    en el sistema."
337         )
338 
339         messagebox.showinfo(
340             "Correcto",
341             "Estudiante agregado"
342         )
343 
344         mostrar_estudiantes()
345         actualizar_comboboxes()
346         limpiar_estudiante()
347 
348     except Exception as e:
349 
350         logging.error(str(e))
351 
352         messagebox.showerror(
353             "Error",
```

```
            str(e)
        )

    finally:

        conn.close()

def seleccionar_estudiante(event):

    selected = tree_estudiantes.focus()

    if not selected:
        return

    valores = tree_estudiantes.item(selected, "values")

    real_id = tree_estudiantes.item(selected, "tags")[0]

    conn = conectar()

    cursor = conn.cursor()

    cursor.execute("""
    SELECT *
    FROM estudiantes
    WHERE id = ?
    """, (
        real_id,
    ))

    row = cursor.fetchone()

    conn.close()

    entry_id.config(state="normal")
    entry_id.delete(0, tk.END)
    entry_id.insert(0, row[0])
    entry_id.config(state="readonly")

    entry_nombre.delete(0, tk.END)
    entry_nombre.insert(0, row[1])

    entry_apellido.delete(0, tk.END)
    entry_apellido.insert(0, row[2])

    entry_email.delete(0, tk.END)
    entry_email.insert(0, row[3])

    fecha = row[4].split("-")
```

```
403
404     combo_anio.set(fecha[0])
405     combo_mes.set(fecha[1])
406     combo_dia.set(fecha[2])
407
408 def actualizar_estudiante():
409
410     if entry_id.get() == "":
411
412         messagebox.showwarning(
413             "Validaci n",
414             "Seleccione estudiante"
415         )
416
417         return
418
419     conn = conectar()
420
421     cursor = conn.cursor()
422
423     cursor.execute("""
424     SELECT *
425     FROM estudiantes
426     WHERE email = ?
427     AND activo = 1
428     AND id != ?
429     """, (
430         entry_email.get(),
431         entry_id.get()
432     ))
433
434     existe = cursor.fetchone()
435
436     if existe:
437
438         messagebox.showwarning(
439             "Validaci n",
440             "Correo ya registrado"
441         )
442
443         conn.close()
444
445         return
446
447     try:
448
449         fecha = f"{combo_anio.get()}-{combo_mes.get()}-{
    combo_dia.get()}"
450
```

```
451         cursor.execute("""
452         UPDATE estudiantes
453         SET nombre=?,
454             apellido=?,
455             email=?,
456             fecha_nacimiento=?
457         WHERE id=?
458         """, (
459             entry_nombre.get(),
460             entry_apellido.get(),
461             entry_email.get(),
462             fecha,
463             entry_id.get()
464         ))
465 
466         conn.commit()
467         crear_backup()
468         log_cambio(
469             "Estudiante",
470             "UPDATE",
471             f"Se actualizo informacion del estudiante ID: {entry_id.get()}",
472             "Los datos actualizados se reflejaran en futuros reportes e inscripciones."
473         )
474 
475         messagebox.showinfo(
476             "Correcto",
477             "Estudiante actualizado"
478         )
479 
480         mostrar_estudiantes()
481         actualizar_comboboxes()
482         limpiar_estudiante()
483 
484     except Exception as e:
485 
486         logging.error(str(e))
487 
488         messagebox.showerror(
489             "Error",
490             str(e)
491         )
492 
493     finally:
494 
495         conn.close()
496 
497 def eliminar_estudiante():
```

```
498
499     if entry_id.get() == "":
500
501         messagebox.showwarning(
502             "Validaci n",
503             "Seleccione estudiante"
504         )
505
506         return
507
508     confirmar = messagebox.askyesno(
509         "Confirmar",
510         " Dar  de baja estudiante?"
511     )
512
513     if not confirmar:
514         return
515
516     conn = conectar()
517
518     cursor = conn.cursor()
519
520     try:
521
522         estudiante_id = entry_id.get()
523
524         cursor.execute("""
525         UPDATE estudiantes
526         SET activo = 0
527         WHERE id = ?
528         """, (
529             estudiante_id,
530         ))
531
532         cursor.execute("""
533         UPDATE inscripciones
534         SET activo = 0
535         WHERE estudiante_id = ?
536         """, (
537             estudiante_id,
538         ))
539
540         conn.commit()
541         crear_backup()
542         log_cambio(
543             "Estudiante",
544             "SOFT_DELETE",
545             f"Se dio de baja al estudiante ID: {estudiante_id
    }",
```

```
            "El estudiante y sus inscripciones ya no apareceran en las listas activas."
        )

        messagebox.showinfo(
            "Correcto",
            "Estudiante dado de baja"
        )

        mostrar_estudiantes()
        mostrar_inscripciones()
        actualizar_comboboxes()
        limpiar_estudiante()

    except Exception as e:

        logging.error(str(e))

        messagebox.showerror(
            "Error",
            str(e)
        )

    finally:

        conn.close()

# ============================================================
# UI ESTUDIANTES
# ============================================================

tk.Label(
    tab_estudiantes,
    text="Gesti n de Estudiantes",
    font=("Segoe UI", 16, "bold"),
    bg="#f3f4f6"
).pack(pady=(10, 0))

tk.Label(
    tab_estudiantes,
    text="Agrega, modifica, elimina y busca informaci n de los alumnos inscritos en el sistema.",
    font=("Segoe UI", 10),
    bg="#f3f4f6"
).pack(pady=(0, 10))

frame_est = tk.Frame(
    tab_estudiantes,
    bg="#f3f4f6"
```

```
)

frame_est.pack(fill="x", padx=10, pady=10)

tk.Label(
    frame_est,
    text="ID",
    bg="#f3f4f6"
).grid(row=0, column=0)

entry_id = tk.Entry(
    frame_est,
    state="readonly"
)

entry_id.grid(row=0, column=1)

tk.Label(
    frame_est,
    text="Nombre",
    bg="#f3f4f6"
).grid(row=1, column=0)

entry_nombre = tk.Entry(frame_est)

entry_nombre.grid(row=1, column=1)

tk.Label(
    frame_est,
    text="Apellido",
    bg="#f3f4f6"
).grid(row=2, column=0)

entry_apellido = tk.Entry(frame_est)

entry_apellido.grid(row=2, column=1)

tk.Label(
    frame_est,
    text="Email",
    bg="#f3f4f6"
).grid(row=3, column=0)

entry_email = tk.Entry(frame_est)

entry_email.grid(row=3, column=1)

# ============================================================
# FECHA
```

```
642 # ============================================================
643
644 tk.Label(
645     frame_est,
646     text="Fecha Nacimiento",
647     bg="#f3f4f6"
648 ).grid(row=4, column=0)
649
650 frame_fecha = tk.Frame(frame_est, bg="#f3f4f6")
651
652 frame_fecha.grid(row=4, column=1)
653
654 combo_dia = ttk.Combobox(
655     frame_fecha,
656     values=[str(i) for i in range(1,32)],
657     width=5
658 )
659
660 combo_dia.set("1")
661
662 combo_dia.pack(side="left")
663
664 combo_mes = ttk.Combobox(
665     frame_fecha,
666     values=[str(i) for i in range(1,13)],
667     width=5
668 )
669
670 combo_mes.set("1")
671
672 combo_mes.pack(side="left", padx=5)
673
674 combo_anio = ttk.Combobox(
675     frame_fecha,
676     values=[str(i) for i in range(1980,2027)],
677     width=8
678 )
679
680 combo_anio.set("2000")
681
682 combo_anio.pack(side="left")
683
684 # ============================================================
685 # BUSCADOR
686 # ============================================================
687
688 tk.Label(
689     frame_est,
690     text="Buscar estudiante por nombre",
```

```
    bg="#f3f4f6",
    font=("Segoe UI", 10, "bold")
).grid(row=5, column=0)

entry_buscar = tk.Entry(frame_est)

entry_buscar.grid(row=5, column=1)

def buscar_estudiante():

    mostrar_estudiantes(
        entry_buscar.get()
    )

tk.Button(
    frame_est,
    text="Buscar",
    bg="#2563eb",
    fg="white",
    command=buscar_estudiante
).grid(row=5, column=2, padx=5)

# ============================================================
# BOTONES
# ============================================================

frame_btn = tk.Frame(
    tab_estudiantes,
    bg="#f3f4f6"
)

frame_btn.pack(fill="x")

tk.Button(
    frame_btn,
    text="Agregar",
    bg="#16a34a",
    fg="white",
    width=15,
    command=insertar_estudiante
).pack(side="left", padx=5)

tk.Button(
    frame_btn,
    text="Actualizar",
    bg="#ca8a04",
    fg="white",
    width=15,
    command=actualizar_estudiante
```

```
).pack(side="left", padx=5)

tk.Button(
    frame_btn,
    text="Eliminar",
    bg="#dc2626",
    fg="white",
    width=15,
    command=eliminar_estudiante
).pack(side="left", padx=5)

tk.Button(
    frame_btn,
    text="Limpiar",
    bg="#6b7280",
    fg="white",
    width=15,
    command=limpiar_estudiante
).pack(side="left", padx=5)

# ============================================================
# TABLA
# ============================================================

tree_estudiantes = ttk.Treeview(
    tab_estudiantes,
    columns=(
        "N",
        "Nombre",
        "Apellido",
        "Email",
        "Nacimiento"
    ),
    show="headings"
)

tree_estudiantes.heading("N", text="#")
tree_estudiantes.heading("Nombre", text="Nombre")
tree_estudiantes.heading("Apellido", text="Apellido")
tree_estudiantes.heading("Email", text="Email")
tree_estudiantes.heading("Nacimiento", text="Nacimiento")

tree_estudiantes.column("N", width=50)

tree_estudiantes.pack(
    fill="both",
    expand=True,
    padx=10,
    pady=10
```

```
)

tree_estudiantes.bind(
    "<<TreeviewSelect>>",
    seleccionar_estudiante
)

# ============================================================
# CURSOS
# ============================================================

def mostrar_cursos():

    tree_cursos.delete(*tree_cursos.get_children())

    conn = conectar()

    cursor = conn.cursor()

    cursor.execute("""
    SELECT *
    FROM cursos
    WHERE activo = 1
    """)

    contador = 1

    for row in cursor.fetchall():

        tree_cursos.insert(
            "",
            "end",
            values=(
                contador,
                row[1],
                row[2]
            ),
            tags=(row[0],)
        )

        contador += 1

    conn.close()

def insertar_curso():

    conn = conectar()

    cursor = conn.cursor()
```

```
838
839     try:
840
841         cursor.execute("""
842         INSERT INTO cursos(
843             nombre,
844             creditos
845         )
846         VALUES(?,?)
847         """, (
848             entry_curso_nombre.get(),
849             entry_creditos.get()
850         ))
851
852         conn.commit()
853         crear_backup()
854         log_cambio(
855             "Curso",
856             "INSERT",
857             f"Se agrego curso: {entry_curso_nombre.get()} ({
    entry_creditos.get()} creditos)",
858             "El curso esta disponible para que los
    estudiantes se inscriban."
859         )
860
861         mostrar_cursos()
862         actualizar_comboboxes()
863
864         messagebox.showinfo(
865             "Correcto",
866             "Curso agregado"
867         )
868
869     except Exception as e:
870
871         messagebox.showerror(
872             "Error",
873             str(e)
874         )
875
876     finally:
877
878         conn.close()
879
880 def seleccionar_curso(event):
881
882     selected = tree_cursos.focus()
883
884     if not selected:
```

```
885         return
886 
887     real_id = tree_cursos.item(selected, "tags")[0]
888 
889     conn = conectar()
890 
891     cursor = conn.cursor()
892 
893     cursor.execute("""
894     SELECT *
895     FROM cursos
896     WHERE id = ?
897     """, (
898         real_id,
899     ))
900 
901     row = cursor.fetchone()
902 
903     conn.close()
904 
905     entry_curso_id.config(state="normal")
906     entry_curso_id.delete(0, tk.END)
907     entry_curso_id.insert(0, row[0])
908     entry_curso_id.config(state="readonly")
909 
910     entry_curso_nombre.delete(0, tk.END)
911     entry_curso_nombre.insert(0, row[1])
912 
913     entry_creditos.delete(0, tk.END)
914     entry_creditos.insert(0, row[2])
915 
916 def eliminar_curso():
917 
918     if entry_curso_id.get() == "":
919         return
920 
921     conn = conectar()
922 
923     cursor = conn.cursor()
924 
925     cursor.execute("""
926     UPDATE cursos
927     SET activo = 0
928     WHERE id = ?
929     """, (
930         entry_curso_id.get(),
931     ))
932 
933     cursor.execute("""
```

```
    UPDATE inscripciones
    SET activo = 0
    WHERE curso_id = ?
    """, (
        entry_curso_id.get(),
    ))

    conn.commit()
    crear_backup()
    log_cambio(
        "Curso",
        "SOFT_DELETE",
        f"Se dio de baja al curso ID: {entry_curso_id.get()}",
        "El curso y las inscripciones asociadas ya no apareceran en las listas activas."
    )

    conn.close()

    mostrar_cursos()
    mostrar_inscripciones()
    actualizar_comboboxes()

# ============================================================
# UI CURSOS
# ============================================================

tk.Label(
    tab_cursos,
    text="Gesti n de Cursos",
    font=("Segoe UI", 16, "bold"),
    bg="#f3f4f6"
).pack(pady=(10, 0))

tk.Label(
    tab_cursos,
    text="Administra el cat logo de cursos, asigna sus nombres y cr ditos correspondientes.",
    font=("Segoe UI", 10),
    bg="#f3f4f6"
).pack(pady=(0, 10))

frame_curso = tk.Frame(tab_cursos, bg="#f3f4f6")

frame_curso.pack(fill="x", padx=10, pady=10)

tk.Label(frame_curso, text="ID", bg="#f3f4f6").grid(row=0, column=0)
```

```
979
980 entry_curso_id = tk.Entry(
981     frame_curso,
982     state="readonly"
983 )
984
985 entry_curso_id.grid(row=0,column=1)
986
987 tk.Label(frame_curso,text="Nombre",bg="#f3f4f6").grid(row=1,
    column=0)
988
989 entry_curso_nombre = tk.Entry(frame_curso)
990
991 entry_curso_nombre.grid(row=1,column=1)
992
993 tk.Label(frame_curso,text="Cr ditos",bg="#f3f4f6").grid(row
    =2,column=0)
994
995 entry_creditos = tk.Entry(frame_curso)
996
997 entry_creditos.grid(row=2,column=1)
998
999 tk.Button(
1000     frame_curso,
1001     text="Agregar Curso",
1002     bg="#16a34a",
1003     fg="white",
1004     command=insertar_curso
1005 ).grid(row=3,column=0,pady=10)
1006
1007 tk.Button(
1008     frame_curso,
1009     text="Eliminar Curso",
1010     bg="#dc2626",
1011     fg="white",
1012     command=eliminar_curso
1013 ).grid(row=3,column=1,pady=10)
1014
1015 tree_cursos = ttk.Treeview(
1016     tab_cursos,
1017     columns=("N","Nombre","Creditos"),
1018     show="headings"
1019 )
1020
1021 tree_cursos.heading("N",text="#")
1022 tree_cursos.heading("Nombre",text="Nombre")
1023 tree_cursos.heading("Creditos",text="Cr ditos")
1024
1025 tree_cursos.pack(fill="both",expand=True,padx=10,pady=10)
```

```
1026
1027 tree_cursos.bind(
1028     "<<TreeviewSelect>>",
1029     seleccionar_curso
1030 )
1031
1032 # ============================================================
1033 # INSCRIPCIONES
1034 # ============================================================
1035
1036 estudiantes_ids = []
1037 cursos_ids = []
1038
1039 def actualizar_comboboxes():
1040     global estudiantes_ids, cursos_ids
1041
1042     conn = conectar()
1043
1044     cursor = conn.cursor()
1045
1046     cursor.execute("""
1047     SELECT id,nombre,apellido
1048     FROM estudiantes
1049     WHERE activo = 1
1050     """)
1051
1052     res_est = cursor.fetchall()
1053     estudiantes_ids = [r[0] for r in res_est]
1054     combo_estudiantes["values"] = [f"{r[1]} {r[2]}" for r in
    res_est]
1055
1056     cursor.execute("""
1057     SELECT id,nombre
1058     FROM cursos
1059     WHERE activo = 1
1060     """)
1061
1062     res_cur = cursor.fetchall()
1063     cursos_ids = [r[0] for r in res_cur]
1064     combo_cursos["values"] = [r[1] for r in res_cur]
1065
1066     conn.close()
1067
1068 def inscribir():
1069
1070     idx_est = combo_estudiantes.current()
1071     idx_cur = combo_cursos.current()
1072
1073     if idx_est == -1 or idx_cur == -1:
```

```
1074            messagebox.showwarning("Validaci n", "Seleccione un
    estudiante y un curso")
1075            return
1076
1077        estudiante_id = estudiantes_ids[idx_est]
1078        curso_id = cursos_ids[idx_cur]
1079
1080        conn = conectar()
1081
1082        cursor = conn.cursor()
1083
1084        cursor.execute("""
1085        SELECT *
1086        FROM inscripciones
1087        WHERE estudiante_id = ?
1088        AND curso_id = ?
1089        AND activo = 1
1090        """, (
1091            estudiante_id,
1092            curso_id
1093        ))
1094
1095        existe = cursor.fetchone()
1096
1097        if existe:
1098
1099            messagebox.showwarning(
1100                "Validaci n",
1101                "El estudiante ya est  inscrito"
1102            )
1103
1104            conn.close()
1105
1106            return
1107
1108        cursor.execute("""
1109        INSERT INTO inscripciones(
1110            estudiante_id,
1111            curso_id,
1112            fecha_inscripcion
1113        )
1114        VALUES(?,?,DATE('now'))
1115        """, (
1116            estudiante_id,
1117            curso_id
1118        ))
1119
1120        conn.commit()
1121        crear_backup()
```

```
1122     log_cambio(
1123         "Inscripcion",
1124         "INSERT",
1125         f"Estudiante ID: {estudiante_id} inscrito en Curso ID
     : {curso_id}",
1126         "Se crea relacion entre estudiante y curso; ambos
     deben estar activos."
1127     )
1128 
1129     conn.close()
1130 
1131     mostrar_inscripciones()
1132 
1133 def mostrar_inscripciones():
1134 
1135     tree_inscripciones.delete(*tree_inscripciones.
     get_children())
1136 
1137     conn = conectar()
1138 
1139     cursor = conn.cursor()
1140 
1141     cursor.execute("""
1142     SELECT
1143         i.id,
1144         e.nombre,
1145         c.nombre,
1146         i.fecha_inscripcion
1147     FROM inscripciones i
1148     JOIN estudiantes e
1149         ON i.estudiante_id = e.id
1150     JOIN cursos c
1151         ON i.curso_id = c.id
1152     WHERE i.activo = 1
1153     AND e.activo = 1
1154     AND c.activo = 1
1155     """)
1156 
1157     contador = 1
1158 
1159     for row in cursor.fetchall():
1160 
1161         tree_inscripciones.insert(
1162             "",
1163             "end",
1164             values=(
1165                 contador,
1166                 row[1],
1167                 row[2],
```

```
                row[3]
            ),
            tags=(row[0],)
        )

        contador += 1

    conn.close()

inscripcion_seleccionada = tk.StringVar()

def seleccionar_inscripcion(event):
    selected = tree_inscripciones.focus()
    if not selected:
        return
    real_id = tree_inscripciones.item(selected, "tags")[0]
    inscripcion_seleccionada.set(real_id)

def cancelar_inscripcion():
    ins_id = inscripcion_seleccionada.get()
    if not ins_id:
        messagebox.showwarning("Validaci n", "Seleccione una
    inscripci n de la lista")
        return

    confirmar = messagebox.askyesno("Confirmar", " Desea  dar
    de baja esta inscripci n?")
    if not confirmar:
        return

    conn = conectar()
    cursor = conn.cursor()
    cursor.execute("UPDATE inscripciones SET activo = 0 WHERE
    id = ?", (ins_id,))
    conn.commit()
    conn.close()

    crear_backup()
    log_cambio(
        "Inscripcion",
        "SOFT_DELETE",
        f"Se cancelo inscripcion ID: {ins_id}",
        "La inscripcion queda inactiva sin eliminar al
   estudiante ni al curso."
    )

    mostrar_inscripciones()
    inscripcion_seleccionada.set("")
    messagebox.showinfo("Correcto", "Inscripci n cancelada")
```

```
1213
1214 def generar_reporte_csv(data):
1215     filepath = filedialog.asksaveasfilename(
1216         defaultextension=".csv",
1217         filetypes=[("CSV Files", "*.csv")],
1218         title="Guardar Reporte"
1219     )
1220     if filepath:
1221         with open(filepath, mode='w', newline='', encoding='
    utf-8') as f:
1222             writer = csv.writer(f)
1223             writer.writerow(["ID Inscripci n", "Nombre
    Estudiante", "Apellido", "Email", "Curso", "Cr ditos", "
    Fecha"])
1224             writer.writerows(data)
1225         messagebox.showinfo("Correcto", "Reporte exportado
    exitosamente")
1226
1227 def abrir_dashboard():
1228     modal = tk.Toplevel(ventana)
1229     modal.title("Dashboard / Reporte General")
1230     modal.geometry("900x500")
1231     modal.configure(bg="#f3f4f6")
1232     modal.grab_set()
1233
1234     tk.Label(modal, text="Reporte General de Inscripciones
    Activas", font=("Segoe UI", 16, "bold"), bg="#f3f4f6").
    pack(pady=10)
1235
1236     # Obtener datos de la base de datos
1237     conn = conectar()
1238     cursor = conn.cursor()
1239     cursor.execute("""
1240     SELECT
1241         i.id,
1242         e.nombre,
1243         e.apellido,
1244         e.email,
1245         c.nombre,
1246         c.creditos,
1247         i.fecha_inscripcion
1248     FROM inscripciones i
1249     JOIN estudiantes e ON i.estudiante_id = e.id
1250     JOIN cursos c ON i.curso_id = c.id
1251     WHERE i.activo = 1 AND e.activo = 1 AND c.activo = 1
1252     """)
1253     data = cursor.fetchall()
1254     conn.close()
1255
```

```
1256     # Configuraci n de la tabla de reporte
1257     tree_reporte = ttk.Treeview(
1258         modal,
1259         columns=("ID", "Estudiante", "Apellido", "Email", "
    Curso", "Creditos", "Fecha"),
1260         show="headings"
1261     )
1262     for col in ("ID", "Estudiante", "Apellido", "Email", "
    Curso", "Creditos", "Fecha"):
1263         tree_reporte.heading(col, text=col)
1264 
1265     tree_reporte.column("ID", width=50)
1266     tree_reporte.column("Creditos", width=80)
1267 
1268     for row in data:
1269         tree_reporte.insert("", "end", values=row)
1270 
1271     tree_reporte.pack(fill="both", expand=True, padx=10, pady
    =10)
1272 
1273     btn_frame = tk.Frame(modal, bg="#f3f4f6")
1274     btn_frame.pack(fill="x", padx=10, pady=10)
1275 
1276     tk.Button(
1277         btn_frame,
1278         text="Descargar CSV",
1279         bg="#16a34a",
1280         fg="white",
1281         font=("Segoe UI", 10, "bold"),
1282         command=lambda: generar_reporte_csv(data)
1283     ).pack(side="right")
1284 
1285     tk.Button(
1286         btn_frame,
1287         text="Cerrar",
1288         bg="#6b7280",
1289         fg="white",
1290         font=("Segoe UI", 10, "bold"),
1291         command=modal.destroy
1292     ).pack(side="right", padx=10)
1293 
1294 # ============================================================
1295 # UI INSCRIPCIONES
1296 # ============================================================
1297 
1298 tk.Label(
1299     tab_inscripciones,
1300     text="Gesti n de Inscripciones",
1301     font=("Segoe UI", 16, "bold"),
```

```
1302     bg="#f3f4f6"
1303 ).pack(pady=(10, 0))
1304 
1305 tk.Label(
1306     tab_inscripciones,
1307     text="Inscribe a los estudiantes en los diferentes cursos
         disponibles de forma r pida y sencilla.",
1308     font=("Segoe UI", 10),
1309     bg="#f3f4f6"
1310 ).pack(pady=(0, 10))
1311 
1312 frame_ins = tk.Frame(tab_inscripciones,bg="#f3f4f6")
1313 
1314 frame_ins.pack(fill="x",padx=10,pady=10)
1315 
1316 tk.Label(frame_ins, text="Seleccionar Estudiante:", bg="#
    f3f4f6").grid(row=0, column=0, padx=5, sticky="w")
1317 tk.Label(frame_ins, text="Seleccionar Curso:", bg="#f3f4f6").
    grid(row=0, column=1, padx=5, sticky="w")
1318 
1319 combo_estudiantes = ttk.Combobox(frame_ins,width=40, state="
    readonly")
1320 
1321 combo_estudiantes.grid(row=1,column=0,padx=5)
1322 
1323 combo_cursos = ttk.Combobox(frame_ins,width=40, state="
    readonly")
1324 
1325 combo_cursos.grid(row=1,column=1,padx=5)
1326 
1327 tk.Button(
1328     frame_ins,
1329     text="Inscribir",
1330     bg="#2563eb",
1331     fg="white",
1332     command=inscribir
1333 ).grid(row=1,column=2,padx=5)
1334 
1335 tk.Button(
1336     frame_ins,
1337     text="Cancelar Inscripci n",
1338     bg="#dc2626",
1339     fg="white",
1340     command=cancelar_inscripcion
1341 ).grid(row=1,column=3,padx=5)
1342 
1343 tree_inscripciones = ttk.Treeview(
1344     tab_inscripciones,
1345     columns=("N","Estudiante","Curso","Fecha"),
```

```
    show="headings"
)

tree_inscripciones.heading("N",text="#")
tree_inscripciones.heading("Estudiante",text="Estudiante")
tree_inscripciones.heading("Curso",text="Curso")
tree_inscripciones.heading("Fecha",text="Fecha")

tree_inscripciones.pack(
    fill="both",
    expand=True,
    padx=10,
    pady=10
)

tree_inscripciones.bind("<<TreeviewSelect>>",
   seleccionar_inscripcion)

# ============================================================
# FOOTER
# ============================================================

firma = tk.Label(
    ventana,
    text="Alejandro Tello Hern ndez | EC0835 | CESUN
   Universidad",
    font=("Segoe UI", 9, "italic"),
    bg="#1f2937",
    fg="white"
)

firma.place(
    relx=1.0,
    rely=1.0,
    anchor="se",
    x=-10,
    y=-25
)

# ============================================================
# DASHBOARD BOTON
# ============================================================

btn_reporte = tk.Button(
    ventana,
    text="       Generar Reporte (Dashboard)",
    bg="#8b5cf6",
    fg="white",
    font=("Segoe UI", 11, "bold"),
```

```
    command=abrir_dashboard
)
btn_reporte.place(relx=0.01, rely=0.98, anchor="sw")

# ============================================================
# INICIO
# ============================================================

mostrar_estudiantes()
mostrar_cursos()
mostrar_inscripciones()
actualizar_comboboxes()

ventana.protocol(
    "WM_DELETE_WINDOW",
    lambda: (
        crear_backup(),
        ventana.destroy()
    )
)

ventana.mainloop()
```