



# Tkinter

## **Documentación de la Aplicación de Gestión de Instituto**

Mercedes Barrera Caro

**Curso:** 2ºDAM

# ÍNDICE

<b>Documentación de la Aplicación de Gestión de Instituto</b> .....	<b>1</b>
1. INTRODUCCIÓN.....	3
2. TECNOLOGÍAS UTILIZADAS.....	4
3. ARQUITECTURA MVC.....	5
4. DISEÑO DE LA BASE DE DATOS.....	6
Tablas principales.....	6
5. FUNCIONALIDADES DE LA APLICACIÓN.....	7
5.1. Pantalla de Login.....	7
5.2. Gestión de Alumnado.....	7
5.3. Gestión de Profesores.....	7
5.4. Gestión de Dirección.....	7
5.5. Gestión de Aulas.....	7
5.6. Gestión de Materiales.....	7
5.7. Gestión de Asignaturas.....	7
5.8. Gestión de Clases.....	7
5.9. Matrículas.....	8
5.10. Convocatorias y Calificaciones.....	8
5.11. Pantalla de Calificaciones.....	8
5.12. Uso de TreeView en las Vistas.....	8
5.13. Exportación de Calificaciones.....	8
6. VALIDACIONES Y CONTROL DE ERRORES.....	9
7. CONTROL DE VERSIONES.....	10
<b>8. CONCLUSIÓN</b> .....	<b>11</b>

# 1. INTRODUCCIÓN

La presente aplicación ha sido desarrollada en Python utilizando Custom Tkinter y sigue el patrón de diseño MVC (Modelo–Vista–Controlador). Está orientada a la gestión administrativa de un Instituto y permite administrar personas, recursos académicos y calificaciones del alumnado.

La aplicación utiliza una base de datos SQLite3 para el almacenamiento persistente de la información y contempla validaciones y control de errores para garantizar la integridad de los datos.

## 2. TECNOLOGÍAS UTILIZADAS

- **Python 3:** Lenguaje de programación principal.
- **Custom Tkinter:** Librería para la creación de interfaces gráficas modernas.
- **SQLite3:** Sistema gestor de base de datos relacional.
- **CSV:** Formato utilizado para la importación y exportación de datos.

### 3. ARQUITECTURA MVC

La aplicación sigue el patrón (Modelo, Vista, Controlador)r, separando claramente las responsabilidades:

- **Modelo:** Representa las entidades del sistema (Alumno, Profesor, Aula, Material, Asignatura, Clase, Matrícula, Dirección, etc.). No accede directamente a la base de datos.
- **Controlador:** Gestiona la lógica de negocio, validaciones y acceso a la base de datos mediante consultas SQL.
- **Vista:** Se encarga exclusivamente de la interacción con el usuario y la presentación de los datos.

Esta arquitectura facilita el mantenimiento del código y cumple los requisitos del enunciado.

## 4. DISEÑO DE LA BASE DE DATOS

La base de datos está diseñada cumpliendo la Tercera Forma Normal (3FN), evitando duplicidades y asegurando la coherencia de la información.

### Tablas principales

- **Alumnos:** Datos del alumnado.
- **Profesores:** Datos del profesorado.
- **Dirección:** Relaciona profesores con cargos directivos (director, jefe de estudios, secretario).
- **Aulas:** Información de las aulas (número, capacidad).
- **Materiales:** Materiales asociados a un aula.
- **Asignaturas:** Nombre y departamento.
- **Clases:** Relaciona profesor, asignatura, aula y año académico.
- **Matrículas:** Relación entre alumno, clase y año académico.
- **Convocatorias:** Tipos de evaluación (1ª evaluación, 2ª evaluación, final, extraordinaria).
- **Calificaciones:** Nota del alumno por convocatoria, con restricciones de unicidad.

Se han definido claves primarias, claves foráneas y restricciones “UNIQUE” para cumplir las reglas de negocio.

## 5. FUNCIONALIDADES DE LA APLICACIÓN

### 5.1. Pantalla de Login

La aplicación dispone de una pantalla de inicio de sesión que valida las credenciales del usuario administrador antes de acceder al menú principal.

### 5.2. Gestión de Alumnado

Permite realizar operaciones CRUD sobre el alumnado:

- Alta de alumnos.
- Modificación de datos.
- Eliminación de registros.
- Validaciones de campos obligatorios y control de duplicidades (DNI único).

### 5.3. Gestión de Profesores

Permite registrar, modificar y eliminar profesores, incluyendo su departamento.

### 5.4. Gestión de Dirección

Los miembros de dirección son profesores con un cargo específico. Esta información se gestiona en una tabla independiente para evitar duplicar datos y mantener la base de datos normalizada.

### 5.5. Gestión de Aulas

Permite administrar las aulas del centro indicando su número y capacidad.

### 5.6. Gestión de Materiales

Se pueden gestionar los materiales del instituto y asociarlos a un aula concreta. Además, se ha implementado:

- **Importación de materiales desde archivo CSV**, validando los datos y asociándolos correctamente a las aulas.

### 5.7. Gestión de Asignaturas

Permite crear y gestionar asignaturas, indicando su nombre y departamento.

### 5.8. Gestión de Clases

Una clase relaciona:

- Profesor
- Asignatura

- Aula
- Año académico

Un profesor puede impartir varias asignaturas en diferentes años.

## 5.9. Matrículas

- El alumnado puede matricularse en una clase por año académico.
- Se garantiza que no haya matrículas duplicadas.
- Una clase puede tener un número indeterminado de alumnos.

## 5.10. Convocatorias y Calificaciones

- El alumnado puede presentarse una sola vez por convocatoria a una asignatura.
- Las calificaciones están restringidas a valores entre 0 y 10.
- Se utilizan claves únicas para asegurar la integridad de los datos.

## 5.11. Pantalla de Calificaciones

La aplicación incluye una pantalla específica para la visualización de calificaciones:

- Muestra las notas por alumno y año académico.
- Navegación entre alumnos mediante botones de siguiente y anterior.

## 5.12. Uso de TreeView en las Vistas

El TreeView se carga con datos obtenidos desde los controladores, respetando el patrón MVC.

### **Ventajas del TreeView:**

- Visualización clara en columnas.
- Mejor organización de la información.
- Selección de filas completas para operaciones CRUD.
- Cumple el requisito del enunciado de mostrar un datagrid.

## 5.13. Exportación de Calificaciones

Se permite exportar las calificaciones del alumnado de una materia a un archivo CSV que incluye:

- Año académico
- Convocatoria
- Nombre y apellidos del alumno
- Asignatura
- Nota

Los datos se exportan ordenados alfabéticamente.



## 6. VALIDACIONES Y CONTROL DE ERRORES

La aplicación incluye:

- Validación de campos obligatorios.
- Control de errores de base de datos.
- Mensajes informativos para el usuario ante errores o acciones correctas.
- Manejo de excepciones para evitar fallos inesperados.

## 7. CONTROL DE VERSIONES

El código fuente se encuentra en un repositorio GitHub y se ha ido actualizando progresivamente mediante distintos commits, cumpliendo con los requisitos de entrega.

## 8. CONCLUSIÓN

La aplicación desarrollada cumple los requisitos establecidos en el enunciado, implementando correctamente una arquitectura MVC, una base de datos normalizada, operaciones CRUD completas y funcionalidades avanzadas como importación y exportación de datos. El uso de TreeView mejora la experiencia de usuario y aporta una presentación más profesional a la aplicación.

Este proyecto ha permitido aplicar de forma práctica los conocimientos adquiridos sobre bases de datos, programación en Python y desarrollo de interfaces gráficas.