IES ISIDRA DE GUZMÁN

PRÁCTICA DE EVALUACIÓN CONTINUA

(PEC)

AUTOR: NORA BEDOYA GARCIA

PROFESOR: RUTH LOSPITAO

MÓDULO PROFESIONAL: Python

CFGS: (2º)Desarrollo de Aplicaciones Multiplataforma

CURSO 2025/26

Contenido

1. OBJETIVO Y PROPÓSITO DE LA APP.

El propósito principal de este proyecto es el desarrollo de una aplicación de consola robusta y modularizada para la gestión y el análisis del rendimiento académico de un grupo de alumnos. La aplicación, denominada **Agenda Académica de Grupo**, está diseñada para:

- **Centralizar el Registro de Datos:** Permitir el registro de todos los eventos de evaluación (**Tareas** y **Exámenes**) para múltiples alumnos y asignaturas.

- **Asegurar la Integridad de Datos:** Aplicar controles de validación, incluyendo formatos **regex** (para DNI y nombres) y rangos numéricos (para notas), garantizando la consistencia del estado de la aplicación.

- **Proporcionar Análisis de Rendimiento:** Generar estadísticas, como medias granulares (por alumno/asignatura) y rankings de asignaturas (**Mejor/Peor asignatura del grupo**), cumpliendo con los requisitos de cálculo y estadística agregada.

- **Demostrar Modularidad:** Implementar una arquitectura de seis ficheros que separa rigurosamente las responsabilidades de E/S, Lógica Pura, Gestión del Estado y Orquestación, superando el requisito mínimo de la práctica.

Se han organizado los datos en una lista de diccionarios (DATOS_AGENDA = []) como se exigía en los requisitos. Al arrancar la aplicación se cargarán los datos que haya en la memoria (si existe) para poder trabajar con ellos y al salir se guardarán automáticamente, aunque también se podrán guardar y sobrescribir a través de la opción 7 del menú, si se necesita.

2. DISEÑO DE MÓDULOS.

(Arquitectura de capas)

Para cumplir con la modularización y las buenas prácticas, el proyecto se estructura en seis ficheros .py, organizados en capas funcionales. Esta arquitectura asegura que la **Lógica de Negocio** esté completamente aislada de las interacciones del usuario y del sistema operativo, facilitando su testeo y mantenimiento.

Módulo	Tipo de Capa	Responsabilidad	Justificación y Rol
servicios.py	**Lógica Pura / Estado**	Gestión del **Estado Global** (DATOS_AGENDA, INDICE_AGENDA) y las funciones de **Lógica Pura** (CRUD, Cálculos, Persistencia Lógica).	**Motor del Negocio.** Único fichero con acceso directo a las estructuras de datos globales. Completamente libre de input() que se manejan con utilidades.py.
controlador.py	**Orquestación / Control**	Dirige el flujo de ejecución: pide datos al usuario (vía utilidades.py) y llama a la lógica de servicios.py.	**Flujo Central.** Actúa como puente entre la Interfaz y el Motor. Contiene las funciones gestionar_alta(), gestionar_informes(), etc.
utilidades.py	**E/S Consola / Validaciones**	Centraliza toda la interacción con el usuario (petición de datos, validación de formatos) y el formateo de salida (ej. imprimir_tabla()).	**Entrada y Salida.** Incluye las lecturas de las entradas que hace el usuario y gestionar validaciones.
persistencia.py	**E/S Disco**	Gestiona la lectura y escritura de archivos físicos (JSON) usando librerías json y os.	**Acceso a Datos Ciego.** Aísla la lógica de negocio de las operaciones de disco.
colores.py	**Configuración**	Define constantes de color ANSI para mejorar la usabilidad y legibilidad de la salida en consola.	**UX/Estilo.** Constantes compartidas.
main.py	**Arranque**	Define la función main() y el bucle de alto nivel (controlador_menu) para iniciar el programa.	**Punto de Entrada.**

3. CAPTURAS DE EJECUCIÓN.

Se incluyen capturas de las funcionalidades más importantes.

- Cuando arranca la aplicación con el menú principal:

```
================================================
---------- AGENDA ACADÉMICA  ----------
================================================
Si necesitas cargar datos de una sesión anterior
antes de empezar, selecciona la opción 7.
================================================
1) Alta de Tarea/Examen
2) Listado Completo
3) Buscar / Filtrar ítems
4) Editar Puntuación
5) Eliminar Ítem
6) Informes y Estadísticas
7) Guardar / Cargar Datos
0) Salir
------------------------------------------------
Selecciona una opción:
```

- Opción 1 del menú- Alta del alumno/tarea:

```
Selecciona una opción: 1

--- ALTA DE NUEVO ÍTEM ---
DNI Alumno (ej. 12345678A, ENTER para cancelar): 12345678a
Nombre Alumno (ENTER para cancelar): nora
Asignatura (ENTER para cancelar): python
Tipo (Tarea/Examen, ENTER para cancelar): tarea
Descripción (ej. 'PEC 1', ENTER para cancelar): pec 1
Nota (0.0-10.0, ENTER si no tiene nota): 8

Ítem registrado con éxito.
```

- Opción 2 del menú-Listado las altas.

```
Selecciona una opción: 2

--- LISTADO COMPLETO DE ÍTEMS ---
+----+-----------+--------+------------+-------+-------+------+
| id |    dni    | nombre | asignatura |  tipo |  desc | nota |
+----+-----------+--------+------------+-------+-------+------+
| 1  | 12345678a | nora   | PYTHON     | TAREA | pec 1 | 8.0  |
+----+-----------+--------+------------+-------+-------+------+
```

- Opción 3: Buscar / Filtrar Items- Los cuales se usarán los filtros de DNI, Asignatura y si es tarea o examen.

```
Selecciona una opción: 3

--- BÚSQUEDA Y FILTRADO DE ÍTEMS ---
Deja un campo vacío (ENTER) si no quieres filtrar por él.
Filtrar por DNI: 12345678a
Filtrar por Asignatura: python
Filtrar por Tipo (Tarea/Examen): tarea

Resultados encontrados (1):
+----+-----------+--------+------------+-------+-------+------+
| id |    dni    | nombre | asignatura |  tipo |  desc | nota |
+----+-----------+--------+------------+-------+-------+------+
| 1  | 12345678a | nora   | PYTHON     | TAREA | pec 1 | 8.0  |
+----+-----------+--------+------------+-------+-------+------+
```

- Opción 4: Editar la nota de un ítem por el cual se deberá indicar el dni y después indicar que ID de item a modificar.

```
Selecciona una opción: 4

--- EDITAR PUNTUACIÓN (Búsqueda por DNI) ---
DNI del Alumno con nota a editar (ENTER para cancelar): 12345678a

Se encontraron 1 ítems para el DNI '12345678A':
+----+------------+-------+-------+------+
| id | asignatura |  tipo |  desc | nota |
+----+------------+-------+-------+------+
| 1  | PYTHON     | TAREA | pec 1 | 8.0  |
+----+------------+-------+-------+------+
ID del ítem exacto que quieres editar (ENTER para cancelar): 1

 Editando: pec 1 de PYTHON - Nota actual: 8.0
Nueva Puntuación (0.0-10.0, ENTER para cancelar): 7

 Puntuación del ítem 1 actualizada a 7.00.
```

- Opción 5: Eliminar ítem.

```
--- ELIMINAR ÍTEM ---
ID del ítem a eliminar (ENTER para cancelar): 1

Ítem con ID 1 eliminado con éxito.
```

- Opción 6: Informes y estadísticas. – Donde podrás además ver media por Alumno y asignatura o ver la media por asignatura. Además por defecto al salir del submenú siempre te aparecerá las mejor y peor asignatura.

```
Selecciona una opción: 6

--- INFORMES Y ESTADÍSTICAS ---

Conteo de Tareas y Exámenes por Asignatura:
+-----------------+--------+-----------+
|   Asignatura    | Tareas | Exámenes  |
+-----------------+--------+-----------+
| Acceso a datos  | 0      | 1         |
| Python          | 1      | 1         |
+-----------------+--------+-----------+

--- Submenú de Medias ---
1. Ver media por Alumno y Asignatura
2. Ver media General por Asignatura
0. Volver al menú de Informes
Selecciona un cálculo: 0

Volviendo al menú de Informes...

--- Ranking de Asignaturas (Según Media General) ---
Mejor Asignatura: Python (Media: 7.50)
Peor Asignatura: Acceso a datos (Media: 6.00)
```

- Opción 7: Guardar los datos o sobreescribir los datos de memoria actuales (por ejemplo, si el curso a acabado e inicia otro “reseteas” la memoria para nuevos datos)

```
Selecciona una opción: 7

--- Persistencia de Datos ---
1. Guardar datos actuales
2. Sobreescribir datos del archivo (se perderán los datos en memoria)
0. Volver al menú principal
Selecciona una opción:
```

4. CASOS DE PRUEBA.

ID	Área de Prueba	Descripción de la Prueba	Entrada de Prueba	Resultado Esperado	Módulo de Manejo (Justificación)
CP.01	**Manejo de E/S Crítica**	Introduce texto en lugar de un número en el menú principal.	Opción: texto	Muestra un error de tipo y vuelve a pedir la opción. El programa **no falla**.	utilidades.py (pedir_entero_obligatorio)
CP.02	**Validación de Patrones**	**Alta (Opción 1):** Intenta introducir un DNI con el formato incorrecto (solo números).	DNI: 12345678	Muestra un error: "Formato DNI incorrecto." y obliga a reintroducir.	utilidades.py (pedir_cadena_con_patron)
CP.03	**Modificación de Datos No Existentes**	**Edición (Opción 4):** Intenta editar la puntuación de un DNI sin registros.	DNI: 00000000X	Muestra un error claro: "No se encontró ningún registro para el DNI..."	controlador.py (gestionar_editar_puntuacion)
CP.04	**Eliminación de Dato Inexistente**	**Eliminación (Opción 5):** Intenta eliminar un ítem con un ID interno que no existe.	ID: 999	Muestra un error: "No se encontró ningún ítem con ID 999 para eliminar."	servicios.py (eliminar_item_logica usando buscar_por_id)
CP.05	**Cancelación de Flujo**	**Alta (Opción 1):** Entra en el flujo y cancela la operación al introducir el DNI.	DNI: (ENTER)	Muestra el mensaje: "Alta cancelada por el usuario." Vuelve al menú principal.	controlador.py (_obtener_datos_alta y gestionar_alta)

5. CONCLUSIONES.

Finalmente he desarrollado la aplicación que deseaba, con validaciones, con estructuras claras y lógicas que quería. Es verdad que al principio me costó bastante estructurar como guardar los datos, porque estaba pensando en Java y lo tenía como muy claro en Java como lo haría, pero no me cuadraba el cómo hacerlo en Python y tuve alguna complicación para llegar a estructurarlos hasta que llegué ahí y finalmente adapté la lógica de como quería las acciones de alta, buscar, editar o eliminar a tenerlo en los datos en una lista de diccionarios. Por lo demás estoy contenta con el resultado de mi aplicación.