Proyecto: Corrector de Exámenes

1. Briefing

- **Objetivo:** Desarrollar un programa en Python para facilitar la gestión y corrección de exámenes en un entorno educativo. El sistema permitirá añadir clases, corregir exámenes y visualizar los últimos resultados.

- **Funciones Principales:**
    - **Añadir Clase:** Permite registrar nuevas clases con su respectivo número de alumnos.
    - **Corregir Examen:** Facilita la corrección de exámenes, calculando la puntuación de cada alumno.
    - **Ver Últimos Exámenes:** Muestra una lista de alumnos con sus notas más recientes por clase.
    - **Corrección Rápida:** Permite una corrección rápida de exámenes sin necesidad de registrar alumnos o clases.

2. Consideraciones Técnicas

- **Lenguaje:** Python 3.12.6
- **Documentación:** Documentacion
- **Desarrollo:** Enfoque modular y orientado a objetos.
- **Entregables:** Código fuente del programa y documentación técnica.

3. Test corrector

- **Propuesta de Valor:** Automatización y optimización del proceso de corrección de exámenes, reduciendo el tiempo y el esfuerzo necesarios.
- **Funcionalidades Estratégicas:**
    - **Interfaz de Usuario Intuitiva:** Utilización de la librería Rich para ofrecer una interfaz visualmente atractiva y fácil de usar.
    - **Gestión de Clases:** Permite la creación y gestión de clases con alumnos.
    - **Corrección de Exámenes:** Facilita la corrección de exámenes con cálculo automático de notas.

        - **Visualización de Resultados:** Proporciona una vista clara de los resultados de los exámenes por clase.
    - **Tecnología y Beneficios:**
        - Python 3.12.6 librerías Rich y Prompt Toolkit.
        - Interfaz de consola moderna, escalabilidad y personalización.
        - **Beneficios:** Ahorro de tiempo, reducción de errores, mejora de la organización y acceso rápido a los resultados.

4. Análisis DAFO

- **Fortalezas:**
    - Interfaz de usuario atractiva y fácil de usar (Rich).
    - Funcionalidades completas para la gestión y corrección de exámenes.
    - Estructura de datos organizada (diccionarios).
    - Feedback al usuario (mensajes claros, tablas informativas).
- **Debilidades:**
    - Falta de persistencia de datos (sin base de datos ni archivos).
    - Validación de datos mejorable.
    - Arquitectura del código monolítica.
    - Falta de seguridad (sin autenticación ni encriptación).
- **Oportunidades:**
    - Implementación de una base de datos SQL.
    - Desarrollo de una versión web.
    - Expansión de funcionalidades (estadísticas, gestión de profesores).
- **Amenazas:**
    - Competencia con soluciones comerciales más completas.
    - Dependencia de bibliotecas externas.
    - Necesidad de formación para usuarios.
    - Costos de mantenimiento y actualizaciones.

5. Tecnologías Usadas

- Visual Studio Code.
- Python con Rich y Prompt Toolkit.
- Sistema operativo: Windows.

6. Diagrama de Flujo

- Diagrama

7. Código Fuente

https://github.com/Jorgemairena13/Corrector-automatico

Codigo fuente

8. Ideas para Mejorar a Largo Plazo

1. **Persistencia de Datos:**
    - Implementar una base de datos (MSQL, SQLite).
    - Guardar la información en archivos JSON o CSV.
2. **Interfaz Web:**
    - Desarrollar una interfaz web con Flask o Django.
    - Diseñar una interfaz de usuario más intuitiva y accesible.
3. **Seguridad:**
    - Implementar autenticación de usuarios (login/password).
    - Añadir roles y permisos (administrador, profesor, alumno).
4. **Funcionalidades Adicionales:**
    - Generar informes y estadísticas.
    - Gestionar el calendario de exámenes.
    - Exportar resultados a PDF.
5. **Experiencia de Usuario:**
    - Mejorar la validación de datos.
    - Añadir mensajes de ayuda y tutoriales.

9. Conclusión

El Corrector de Exámenes es un sistema desarrollado en Python para facilitar la gestión y corrección de exámenes en un entorno educativo. Gracias a su interfaz visual y sistema de menús intuitivo, el proyecto proporciona una base sólida para futuras mejoras y extensiones.