Trabajo Final para la Cátedra: Estructuras de Datos en Python

Pautas para el Trabajo:

• Los trabajos se desarrollarán en grupos de 2 o 3 personas.
• La conformación del grupo, se deberá respetar hasta el final.
• La fecha de entrega del trabajo es la definida por el Docente. Grupo que no presenta en esa fecha, se le bajarán puntos.

Sobre la Nota Final:

• La nota final del trabajo se evaluará teniendo en cuenta varios aspectos clave. En primer lugar, se considerará la entrega en tiempo y forma. Esto implica que el trabajo debe ser entregado dentro del plazo estipulado y en el formato adecuado, con todos los archivos y documentación solicitados. Cualquier retraso será penalizado de acuerdo con las reglas establecidas previamente.
• Otro criterio importante es la resolución del problema. Se evaluará la correctitud de la solución implementada, valorando el uso adecuado de las estructuras de datos y algoritmos enseñados en clase. Además, se tendrá en cuenta la eficiencia de la implementación, prestando especial atención a la complejidad computacional y el manejo de casos excepcionales o borde.
• La documentación complementaria también tendrá un peso significativo en la evaluación. El material adicional presentado, como presentaciones en PowerPoint o videos explicativos, será analizado en términos de su calidad y claridad. Se espera que estos documentos expongan de manera coherente el problema y las soluciones adoptadas, utilizando gráficos o tablas para respaldar la explicación cuando sea necesario. La documentación debe además justificar las decisiones de diseño y demostrar cómo la solución cumple con los requisitos planteados.
• Por último, la defensa individual de cada estudiante jugará un rol crucial en la nota final. Durante esta etapa, se evaluará la capacidad de cada uno para defender y justificar su solución, mostrando un conocimiento profundo del código implementado y de los conceptos aplicados. También se valorará la habilidad para responder preguntas técnicas relacionadas con la solución, así como la claridad y seguridad en la exposición oral.
• Asimismo, se tendrá en cuenta el uso de buenas prácticas, como la correcta documentación del código, el cumplimiento de convenciones de programación y el manejo de versiones, si corresponde.
• Cualquier duda, consulta deberá coordinarse con los Docentes de la Materia.

> **Importante:**
>
> - Cada entrega parcial será evaluada bajo estos mismos criterios. La calidad y entrega de los materiales adicionales (presentaciones, infografías, videos, síntesis) es obligatoria en cada etapa, no solo en la entrega final.
> - La defensa oral del trabajo será únicamente en la última entrega.
> - Todo el material (código, documentación y adicionales) debe ser subido a un repositorio de GitHub, cuyo enlace se entregará en cada entrega.

Trabajo Final para la Cátedra: Estructuras de Datos en Python

Tema: Cliente de Correo Electrónico (Email Client)

Pautas para el Trabajo:

- Los trabajos se desarrollarán en grupos de 2 o 3 personas.
- La conformación del grupo se deberá respetar hasta el final.
- La fecha de cada entrega parcial y la entrega final será definida por el Docente.
- Cada entrega parcial será evaluada bajo los mismos criterios.
- Los materiales adicionales (presentaciones, infografías, videos, síntesis) son obligatorios en cada etapa.
- La defensa oral será únicamente en la última entrega.
- Todo el trabajo y materiales deben subirse a un repositorio de GitHub y entregar el enlace en cada entrega.
- Cualquier duda, consulta deberá coordinarse con los Docentes de la Materia.

Sobre la Nota Final:

- Se evaluará la entrega en tiempo y forma, la correctitud y eficiencia de la solución, el uso adecuado de estructuras de datos y algoritmos, la calidad de la documentación y materiales adicionales, la defensa oral final, y las buenas prácticas de programación y manejo de versiones.

Hilo Conductor: Cliente de Correo Electrónico

El objetivo es diseñar e implementar, en Python, un sistema orientado a objetos que modele un cliente de correo electrónico, permitiendo la gestión de usuarios, mensajes, carpetas, filtros y operaciones típicas de un entorno de email.

Entregas Parciales (cada 3 semanas):

Entrega 1: Modelado de Clases y Encapsulamiento

- Definir las clases principales: Usuario, Mensaje, Carpeta, ServidorCorreo.
- Encapsular atributos y métodos, utilizando propiedades y métodos de acceso.
- Implementar interfaces para enviar, recibir y listar mensajes.
- Documentar el diseño con diagramas de clases y justificación de decisiones.

Entrega 2: Estructuras de Datos y Recursividad

- Implementar la gestión de carpetas y subcarpetas como una estructura recursiva (árbol general).
- Permitir mover mensajes entre carpetas y búsquedas recursivas de mensajes por asunto/remitente.
- Analizar la eficiencia de las operaciones implementadas.
- Material adicional: infografía o video explicativo del árbol de carpetas.

Entrega 3: Algoritmos y Funcionalidades Avanzadas

- Implementar filtros automáticos usando listas y diccionarios para organizar mensajes según reglas definidas por el usuario.
- Agregar una cola de prioridades para gestionar mensajes marcados como “urgentes”.
- Modelar la red de servidores de correo como un grafo (nodos: servidores, aristas: conexiones) y simular el envío de mensajes entre servidores usando BFS/DFS.
- Material adicional: presentación o síntesis de los algoritmos utilizados.

Entrega 4: Integración, Presentación y Defensa

- Integrar todas las funcionalidades en una interfaz de línea de comandos o simple GUI.
- Documentar el código y justificar las decisiones de diseño orientadas a objetos.
- Subir todo el proyecto y materiales al repositorio de GitHub.
- Defensa oral grupal e individual (solo en esta entrega), donde cada integrante debe explicar y justificar partes del código y responder preguntas técnicas.

Criterios de Evaluación

- **Entrega en tiempo y forma** de cada parcial y del final.
- **Correctitud y eficiencia** de la solución, uso adecuado de OOP, estructuras de datos y algoritmos.
- **Calidad y claridad** de la documentación y materiales adicionales en cada entrega.
- **Justificación de decisiones de diseño** y análisis de eficiencia.
- **Defensa oral** (solo en la última entrega): conocimiento profundo del código y conceptos aplicados.
- **Buenas prácticas**: documentación del código, convenciones, manejo de versiones (GitHub).
- **Subida obligatoria** de todo el trabajo y materiales al repositorio de GitHub.

¿Dudas o consultas? Contactar a los Docentes