Hemos planteado las siguientes ideas:

1. Utilizaremos Github, el cual nos permitirá trabajar simultáneamente en el código.
2. Las reuniones las haremos presencialmente y vía online por medio de programas como AnyDesk (para compartir la pantalla y poder controlar lo que se hace por cada uno de las personas conectadas) y llamadas por WhatsApp.
3. Se utilizarán estructuras de datos como listas enlazadas y grados. También implementaremos JSON para el manejo de entradas y salidas.
4. Utilizaremos librerías que nos ayudarán en el desarrollo del proyecto como matplotlib y networkX.
5. Se creará un sistema que simule un gestor de archivos, donde se podrá generar archivos, directorios, subdirectorios, utilizando JSON.
6. Para que el programa sea perpetuo, es decir, no finalice hasta que el usuario indique el cierre del mismo, el programa se correrá en un ciclo infinito.
7. El programa recibirá, por parte del usuario, los parámetros pertinentes para darle flujo al programa, dichos parámetros se recibirán en primera instancia como una cadena, la cual será convertirá en un arreglo cuyos índices serán dichos parámetros.
8. Con el arreglo creado de la cadena que introduzca el usuario, se validarán los comandos, verificando la cantidad de parámetros que ha introducido, por ejemplo, si el índice 0 del arreglo es “ln” y el 1 es “/home” el comando será válido, si se detecta más índices, será un comando inválido.
9. Los comandos válidos que procesará el programa son:
    a. ls
    b. ls -1
    c. mkdir [NOMBRE DEL DIRECTORIO]
    d. help
    e. plot
    f. ln [RUTA DEL ARCHIVO]
    g. touch [NOMBRE DEL ARCHIVO]
    h. rm [NOMBRE DEL ARCHIVO]
    i. rmdir [NOMBRE DEL DIRECTORIO]
    j. cd [NOMBRE DEL DIRECTORIO]
    k. cd ..
    l. findbe [EXTENSIÓN DEL ARCHIVO]
    m. exit
10. Para los comandos “ls” y “ls -1” se recuperarán los registros en el grafo y se formatearan para generar una tabla que será mostrada al usuario en la consola.

11. Para generar un enlace se debe ingresar, como una sola cadena, la ruta del archivo junto a su nombre y no el nombre mismo.
12. Para la navegación entre directorios y subdirectorios en el árbol, de manera conveniente, para una mayor facilidad en flujo del programa, se almacenará en un arreglo, las rutas solicitadas por el usuario, de esta manera se podrá acceder a cada uno de ellos con mayor facilidad y los comandos soportados por el sistema gozaran de más eficiencia.
13. El comando touch debe recibir, obligatoriamente, un nombre de archivo con su extensión.
14. Las carpetas no deben contener puntos en sus nombres.
15. Se agregará, como extra, un comando adicional, la instrucción “exit”, la cual romperá de inmediato la ejecución del programa.
16. Cada uno de los comandos se deberán correr sobre la consola de Ubuntu, dentro de la ejecución del programa de manera perpetua.
17. Cada uno de los comandos deberá estar debidamente blindado, para no generar errores internos que puedan romper la ejecución del programa mismo.
18. El sistema deberá ser capaz de procesar archivos de manera interna solo para la estructura JSON.
19. La estructura JSON se guardará en un archivo, así el programa podrá leer y escribir en él, en todo momento.
20. Al iniciar el programa se debe cargar el archivo con la estructura JSON, luego, deberá leer su contenido para crear un grafo con dicha información, se debe hacer el proceso inverso para guardar la información actual en el archivo.
21. El diseño de la interfaz de usuario del programa consistirá en:
    a. Crear un nombre y logo con código ASCII para que el programa sea agradable para el usuario.
    b. Separadores en cada una de sus partes.
    c. Símbolo del dólar para indicarle al usuario el ingreso de un comando.
    d. La ruta antes del símbolo del dólar para indicarle al usuario en que directorio está ubicado actualmente.
    e. Mensajes de error al usuario indicándole los pasos a seguir para darle el flujo correcto al programa.
22. El programa al iniciarse deberá mostrar un mensaje como cabecera, el cual contendrá:
    a. Logo del programa (código ASCII).
    b. El nombre del programa.
    c. Programadores del proyecto.
    d. La versión del mismo.
    e. Descripción breve del programa.
23. El sistema deberá realizar las cuatro operaciones básicas:
    a. Crear archivos, carpetas y enlaces a archivos.

b. Leer rutas, grafos, archivo en disco duro, registros de archivos y carpetas.
c. Actualizar ruta, archivo o carpeta.
d. Eliminar ruta, archivo o carpeta.

24. Todo procedimiento debe trabajarse sobre el grafo.

25. Se tomarán fotografías y se creará un video que se proveerán como evidencia del trabajo realizado. Todo ello ira en archivos pdfs individuales, para luego ser enviadas para su posterior revisión.

26. Se utilizarán TDAs para manejar los procedimientos internos del programa.

27. Se implementará el uso de captura de fecha y hora para registrar las acciones del programa.

28. Se pondrá en practica las clases ya conocidas y programadas en la asignatura para generar listas enlazadas (LinkedList.py), grafos (Graph.py), nodos (Node.py), papelera de reciclaje (Trash.py)

29. Solamente se creará una ventana (interfaz gráfica) que se usará para embeber la imagen del grafo diseccionado en la pantalla del usuario, cuando este ejecute el comando “plot”.

30. El programa se codificará en idioma inglés con comentarios y salidas de pantalla en español.

31. El programa debe ser capaz de procesar caracteres latinos.

32. Implementaremos un alfabeto como guía para balancear (ordenar) el grafo, se comparará cada uno de los nombres de los enlaces, archivos y carpetas para ordenarlos alfabéticamente; evitando así usar los operadores lógicos usuales “<” y “>” para la comparación de las cadenas.

33. Para futuras revisiones y entendimiento del código generado por parte de docentes o programadores, se comentará el código a modo de explicación de las partes fundamentales del mismo, así se podrá dar mantenimiento más efectivo y eficiente del mismo.

34. De manera interna, se generará un archivo CSV donde se guardará el contenido del grafo en tiempo real, esto servirá para mostrar que ocurre con el grafo en todo momento.