Cloner proporciona respaldo de archivos "versionados"; esto quiere decir que cada archivo respaldado por Cloner tiene asociadas varias versiones (cambios). Cada una de estas versiones cuenta con un momento en el tiempo (timestamp) desde el cual el archivo se considera válido (última modificación) y un instante en el que deja de serlo (porque se ha agregado una versión más reciente, o porque el archivo ha sido eliminado del equipo del usuario).

La tarea es implementar una estructura de datos y sus métodos que permitan lo siguiente:

- **Agregar una versión de un archivo (que podría o no existir previamente)**
- AddFile(path string, created time.Time)
- **Agregar un directorio (que podría o no existir previamente)**
- AddFolder(path string, created time.Time)
- **Notificar de la invalidación o eliminación de una versión**
- DeleteFile(path string, deleted time.Time)
- **Notificar de la invalidación de un directorio, de sus contenidos, y de todas sus ramas de archivos (archivos "hijos").**
- DeleteFolder(path string, deleted time.Time)
- **Consultar si un archivo existía (o tenia una versión válida), para la fecha proporcionada.**
- Exists(path string, when time.Time) bool
- **Listar el contenido de un directorio en una fecha determinada. Es necesario que la lista de archivos retornada se encuentre ordenada por nombre (en orden lexicográfico).**
- ListDir(path string, when time.Time) []string
- **Dado un intervalo entre dos fechas, retornar el listado de versiones válidas que existen para el archivo (ordenados por fecha decreciente).**
- VersionList(path string, start, end time.Time) []Version

* Ponderaremos positivamente una menor complejidad algorítmica a pesar de un mayor uso de memoria.
* Los métodos propuestos son sólo una sugerencia, implementarlos a criterio personal
* Los directorios vacíos (sin archivos) no es necesario retornarlos.
* La firma de las funciones se encuentra en Golang, si gusta puede responder en otro lenguaje a elección.