04 POR HACER

FUNCIONALIDAD DE LOS PAQUETES

**app-01-IdCmd.js**

* Este pck es ejecutado desde la consola y al ejecutarlo deben ser enviados dos argumentos: ‘comando’ y ‘descripcion’ (el comando es una palabra sin guion y los datos una letra con doble guion o una palabra con guion); estos argumentos se graban en el objeto ‘yargs’ (predefinido en node)
* Hace un ‘require’ al pck ‘yargs-u.js’ (hecho por el usuario) ubicado en el folder ‘config’ para que identifique y valide los argumentos recibidos desde la consola y en caso de ser validos los asigne a la constante ‘argv’.
* Dependiendo del comando recibido en argv.\_[0], hace una eleccion con la instrucción ‘switch’ para hacer alguna tarea. Si no se recibe ningun comando de los definidos en ‘yargs-u.js’ .. se envia mensaje indicandolo

**app-02-PorHacer.js**

* Este pck es ejecutado desde la consola y al ejecutarlo deben ser enviados los argumentos ‘comando’ y ‘descripcion’ (el ‘comando’ es una palabra sin guion y la descripcion un guion y una letra (alias) seguida del texto o dos guiones y una palabra seguido del texto); estos argumentos se graban en el objeto ‘yargs’ (predefinido en node)
* Hace un ‘require’ del pck ‘yargs-u.js’ (hecho por el usuario) ubicado en el folder ‘/config’ para recibir los datos enviados desde la consola los cuales asigna a la constante ‘argv’; el ‘yargs-u.js’ identifica y valida el ‘comando’ y la ‘descripcion’ recibidos desde la consola y los asigna a variables que podran ser usadas posteriormente durante el proceso.
* Hace un ‘require’ del pck ‘colors’ (predefinido en node)
* Hace un ‘require’ del pck ‘por-hacer’ (hecho por el usuario) ubicado en el folder ‘/por-hacer’ y lo asigna a la constante ‘porHacer’ de manera que se puedan invocar las funciones contenidas en dicho pck ‘por-hacer’
* Declara la variable ‘comando’ y le asigna el contenido de ‘argv.\_[0] recibido de ‘yargs-u’
* Con la instrucción ‘switch’ prueba el contenido de la variable ‘comando’ y hace una eleccion para ejectuar una tarea. Si no se recibe ningun comando de los definidos en ‘yargs-u.js’ .. se envia mensaje indicandolo
  + Si el comando recibido de la consola es ‘crear’ invoca a la funcion ‘porHacer.crear’ a la cual le envia la ‘descripcion’ en la variable ‘argv.descripcion’ la cual graba en la tabla ‘data.json’, la ‘descripcion’ y el ‘completado’ (en staus ‘falso’) y retorna lo grabado en el arreglo ‘listadoPorHacer’ el cual asigna a la variable ‘tarea
  + Si el comando recibido de la consola es ‘listar’ invoca a la funcion ‘porHacer.getListado’ sin enviarle ningun argumento. Dicha funcion lee el contenido de la tabla ‘data.json’ y lo retorna en el arreglo ‘listadoPorHacer’ el cual se asigna al objeto ‘listado’ ; con un comando ‘for’ se asigna el contenido del objeto ‘listado’ al objeto ‘tarea’ y se publica con colores la ‘descripcion’ y el ‘estado’.
  + Si el comando recibido de la consola es ‘actualizar’, le asigna a la variable ‘actualizado’ el valor retornado al invocar la funcion ‘porHacer.actualizar’ enviandole los argumentos ‘argv.descripcion’, ‘argv.completado’; publica la variable ‘actualizado’
  + Si el comando recibido de la consola es ‘borrar’, le asigna a la variable ‘borrado’ el valor retornado al invocar la funcion ‘porHacer.borrar’ enviandole el argumento ‘argv.descripcion’

**Yargs-u.js**

* Este pck esta ubicado en el folder ‘config’
* Es llamado por un pck principal
* Su objetivo es con el fin de que tome los datos grabados en el objeto ‘yargs’ , los cuales son recibidos desde la consola cuando es ejecutado el pck principal
* Hace un ‘require’ del objeto ‘yargs’ y lo asigna a la constante argv; en el ‘yargs’ lee los argumentos recibidos de la consola al momento que se ejectuta el pck principal
* los argumentos recibidos sin guion seran guardados como primer dato en ‘yargs’ y los argumentos con guion y una letra o dos guiones y una palabra seran guardados como siguientes datos en ‘yargs’
* El argumento guardado como primer dato en ‘yargs’ sera identificado como el ‘command’
* Los argumentos guardados como siguientes datos seran identificados como la ‘descripcion’ y ‘completado’
* Dichos datos se guardaran en la constante ‘argv’ la cual se exportara al pck principal
* Todas estas acciones podrian hacerse en el pck principal pero es recomendable hacerlo como un sub pck para simplificar el codigo del pck principal y ademas no estarlo repitiendo cuando cualquier otro pck que lo requiera.
* Simplificacion: Antes de la declaracion de ‘argv’ se declaran los objetos ‘descripcion’ y ‘completado’ con todos sus parametros; en las declaraciones ‘command’ dentro de ‘argv’ solo se incluye la palabra ‘descripcion’ y la palabra ‘completado’; esto evita declarar varias veces los dos objetos con todos sus parametros.

**Por-hacer.js**

* Este pck esta ubicado en el folder ‘por-hacer’ y es llamado desde otro pck principal
* Su objetivo es leer y grabar datos persistentes en la tabla ‘data.json’ ubicada en el folder ‘db’
* Recibe del pck principal como argumento una ‘descripcion’ la cual debio ser enviada desde la consola al ejecutarse el pck principal
* Hace un ‘require’ del objeto ‘fs’ (natural de node) y lo asigna a la constante ‘fs’
* Declara un arreglo global ‘ListadoPorHacer[]’
* Declara una funcion de flecha ‘cargarDB’ la cual no recibe ningun argumento; hace un ‘require’ de la tabla ‘../db/data.json’ y asigna su contenido al arreglo ‘listadoPorHacer’
* Declara una funcion de flecha ‘guardarDB’ la cual no recibe ningun argumento; asigna el contenido del objeto ‘listadoPorHacer’ a la variable ‘data’ y la graba en la tabla ‘data.json’ invocando la funcion ‘fs.writeFile’
* Declara la funcion de flecha ‘crear’ la cual recibe el argumento ‘descripcion’; invoca a la funcion ‘cargarDB’ para leer el contenido de la tabla ‘data.json’; con la ‘descripcion’ recibida foma el objeto ‘porHacer’ y lo adiciona al arreglo ‘ListadoPorHacer’; invoca la funcion ‘guardarDB()’ para guardar ‘listadoPorHacer’ en la tabla ‘data.json’; retorna el objeto ‘porHacer’ al programa principal
* Declara la funcion de flecha ‘getListado’ la cual no recibe ningun argumento; invoca a la funcion ‘cargarDB’ para leer el contenido de la tabla ‘data.json’ el cual es retornado en el arreglo ‘listadoPorHacer’ y dicho arreglo es retornado al pck llamador.
* Declara la funcion de flecha ‘actualizar’ la cual es llamada desde otro pck y recibe dos argumentos: ‘descripcion’ y ‘completado’ este ultimo se declara como ‘true’ por default al declarar los argumentos; invoca a la funcion ‘cargarDB()’ para leer el contenido de la tabla ‘data.json’ el cual es retornado en el arreglo ‘listadoPorHacer’; en dicho arreglo busca con ‘findIndex’ el indice del elemento recibido en ‘descripcion’ y lo asigna a la variable ‘index’; prueba si el ‘index’ es mayor o igual a cero con una funcion ‘if’; si el resultado de la prueba es verdadero le asigna a ese elemento el valor de ‘completado’ recibido, invoca a la funcion ‘guardarDB()’ para grabar el nuevo dato, y retorna un ‘true’; si el resultado de la prueba es falso, no hace nada y retorna un ‘false’
* Declara la funcion de flecha ‘borrar’ la cual es llamada desde otro pck; recibe el argumento ‘descripcion’; invoca la funcion ‘cargarDB()’ para leer el contenido de la tabla ‘data.json’ el cual es retornado en el arreglo ‘listadoPorHacer’; se declara la variable ‘nuevoListado’ a la cual se le asigna el contenido del arreglo ‘listadoPorHacer’ despues de haberle quitado el elemento que sea igual al argumento ‘descripcion’ utilizando la funcion ‘filter’; se prueba si se quito algun elemento con una funcion ‘if’ comparando las longitudes de ‘lilstadoPorHacer’ con ‘nuevoListado’ con la funcion ‘lenght’; si la prueba es falsa se retorna un ‘false’ al pck llamador; si la prueba es verdadera se asigna el contenido del arreglo ‘nuevoListado’ al arreglo ‘listadoPorHacer’ ya sin el elemento removido; se invoca la funcion ‘guardarDB()’ para grabar el nuevo arreglo a la tabla ‘dara.json’ y se retorna un ‘true’ al programa llamador.
* En la funcion ‘module.exports’ del pck ‘module’ (natural de node) se incluyen las funciones ‘crear’, ‘getListado’, ‘actualizar’.