Module client

Function Index

client/1	Dada la estructura con la informacion de los servidores memcached corriendo, recibe los pedidos del cliente y los responde indefinidamente, o hasta que se haga un pedido de cierre.
del/2	Elimina el par clave-valor asociado a la clave objetivo en la instancia de cliente con el PID pasado.
get/2	Obtiene el valor asociado a la clave objetivo en la instancia de cliente con el PID pasado.
put/3	Almacena el par clave-valor indicado en el servidor memcached asociado a la instancia de cliente con el PID pasado.
quit/1	Finaliza la ejecucion de cierto cliente.
start/1	Toma una lista de pares IP y puerto correspondientes a servidores memcached, crea el socket para la conexion con cada uno de ellos, y genera el proceso cliente que se utilizara para hacer los pedidos.
startDefault/0	Inicializa el proceso cliente con valores establecidos por defecto.
stats/1	Obtiene e imprime las estadisticas de los servidores a los que el cliente con el PID pasado se encuentra conectado.
status/1	Obtiene e imprime de las estadisticas que lleva localmente el cliente con el PID pasado.

Function Details

client/1

```
client(ServersTable::servers_table()) -> term()
```

`ServersTable`: La estructura con la informacion necesario para que el cliente pueda ejecutarse.

Dada la estructura con la informacion de los servidores memcached corriendo, recibe los pedidos del cliente y los responde indefinidamente, o hasta que se haga un pedido de cierre.

del/2

```
del(ClientPID::pid(), Key::term()) -> term() | nonExistingClient
```

`ClientPID`: el PID del cliente al cual le queremos hacer una operacion de DEL.
`Key`: La clave del par a eliminar.

returns: La respuesta del cliente o nonExistingClient si no existe un cliente asociado al PID pasado.

Elimina el par clave-valor asociado a la clave objetivo en la instancia de cliente con el PID pasado.

get/2

```
get(ClientPID::pid(), Key::term()) -> term() | nonExistingClient
```

`ClientPID`: el PID del cliente al cual le queremos hacer una operacion de GET.
`Key`: La clave del par a buscar.

returns: La respuesta del cliente o nonExistingClient si no existe un cliente asociado al PID pasado.

Obtiene el valor asociado a la clave objetivo en la instancia de cliente con el PID pasado.

put/3

```
put(ClientPID::pid(), Key::term(), Value::term()) -> term() | nonExistingClient
```

`ClientPID`: el PID del cliente al cual le queremos hacer una operacion de PUT.
`Key`: La clave a almacenar.
`Value`: El valor asociado a la clave.

returns: La respuesta del cliente o nonExistingClient si no existe un cliente asociado al PID pasado.

Almacena el par clave-valor indicado en el servidor memcached asociado a la instancia de cliente con el PID pasado.

quit/1

```
quit(ClientPID::pid()) -> clientClosed | nonExistingClient
```

`ClientPID`: el PID del cliente que queremos cerrar.

returns: clientClosed indicando que se cerro el cliente o nonExistingClient si no existe un cliente con el PID pasado.

Finaliza la ejecucion de cierto cliente.

start/1

```
start(ServerList::[util:server_info()]) -> {invalidServer, server_info()} | pid()
```

`ServerList`: una lista de tuplas {IP, puerto} cada una correspondiente a un servidor memcached.

returns: El PID del nuevo cliente creado y que sera usado por las demas funciones.

Toma una lista de pares IP y puerto correspondientes a servidores memcached, crea el socket para la conexion con cada uno de ellos, y genera el proceso cliente que se utilizara para hacer los pedidos.

startDefault/0

```
startDefault() -> existingClient | clientCreated
```

returns: existingClient si ya se habia registrado a un cliente previamente, clientCreated si no se habia registrado previamente y se logro lanzar el proceso cliente exitosamente.

Inicializa el proceso cliente con valores establecidos por defecto. Considera funcionando a dos servidores en la direccion de loopback de IPv4 en los puertos 8000 y 9000.

stats/1

```
stats(ClientPID::pid()) -> ok | nonExistingClient
```

`ClientPID`: el PID del cliente al cual le queremos hacer una operacion de STATS.

returns: ok o nonExisting client si no existe un cliente asociado al PID pasado.

Obtiene e imprime las estadisticas de los servidores a los que el cliente con el PID pasado se encuentra conectado.

status/1

```
status(ClientPID::pid()) -> ok | nonExistingClient
```

`ClientPID`: el PID del cliente al cual le queremos hacer una operacion de STATUS.

returns: ok o nonExisting client si no existe un cliente asociado al PID pasado.

Obtiene e imprime de las estadisticas que lleva localmente el cliente con el PID pasado.
