# Ejercicio Evaluable 3

***RPC***

**Sistemas Distribuidos**

**Alfonso Pineda (100472157)**

Course 2023/2024

Degree in Computer Science and Engineering

# Tabla de Contenidos

## Diseño no Distribuido

La parte del diseño no distribuido consta de las funciones que son especificadas en el enunciado del ejercicio (init, set_value, get_value, modify_value, delete_key y exists), igual que en el anterior ejercicio, yo decidí utilizar archivos de texto dentro de un directorio llamado “BASEDEDATOS” para guardar las tuplas del programa, esta decisión la tomé porque me resultó más fácil, ya que la implementación la hice enteramente yo (la diseñe e implementé desde 0) en el anterior ejercicio evaluable.

No le hice ningún cambio a esta parte del programa.

## Diseño Distribuido y RPC

Esta parte del programa me costó más de lo necesario, ya que no estaba familiarizado con los archivos .x y el comando rpcgen que crea una estructura para el proyecto.

Después de saber cómo funcionan los archivos .x y rpcgen intenté descartar por completo la parte modificable de la estructura, perdiendo tiempo porque al intentar hacer todo lo que rpcgen hacía por mi cuenta (y también he de destacar que no es fácil encontrar información sobre RPC con .x y c) inevitablemente iba a fallar.

Tuve que probar varias veces con varios .x para llegar a una versión que pudiera usar para continuar el programa, que no fue la versión final. Con esta versión pude hacer la mayoría de funciones requeridas por el cliente (todas excepto get).

Pude solucionar el problema que tenía con la función get porque hice un nuevo .x para optimizar y simplificar funciones y de alguna manera el problema de la función get desapareció y ahora todas las funciones cumplen los requisitos.

El repositorio del profesor que suelo usar de referencia “solo” me sirvió para saber que tenía que usar el comando rpcgen, archivos .x y que puedes crear vectores típicos de cpp con rpcgen, el resto del ejercicio lo descubrí en su gran mayoría por mi cuenta.

## Concurrencia

En este ejercicio no usé concurrencia porque no encontré información sobre cómo implementarla ya que no tenía suficiente tiempo para probar e implementar algo desde 0 y no hay mucha información disponible sobre la tecnología en cuestión.

## Forma de compilar y ejecutar

El archivo comprimido y el makefile están hechos para que la ejecución del programa sea tan fácil como descomprimir el archivo tar, ejecutar el comando make en consola y ejecutar servidor y cliente (en ese orden). Dependiendo si se utiliza un sistema operativo distinto a Ubuntu LTS 22.04 LTS (ese es el que yo utilizo) hay que comentar las líneas del makefile que añaden flags que no son necesarios (y no sé hasta qué punto pueden perjudicar el compilado en otros sistemas), En ese caso se debería comentar la línea 24 y 26 del makefile y descomentar 25 y 27 del mismo.

## Testing

Las pruebas que he hecho al programa fueron distintas combinaciones con diferentes estados (de la “BASEDEDATOS”) comprobando la rigidez del programa, ya que he probado prácticamente todas las combinaciones de resultados.

## Conclusión

Aprendí bastante con esta práctica, tanto en desarrollo de software (con código ya establecido, en el sentido de desarrollar software en un programa que ya tiene sus bases hechas, tuve que leer mucha documentación, probar mucho, romper el código, arreglarlo y aprender cómo funcionaba para poder modificarlo, algo bastante habitual en casos reales de compañías grandes con software ya establecido, aunque la escala no sea para nada proporcional) como en sistemas distribuidos, ya que pude probar distintas formas de utilizar rpc.