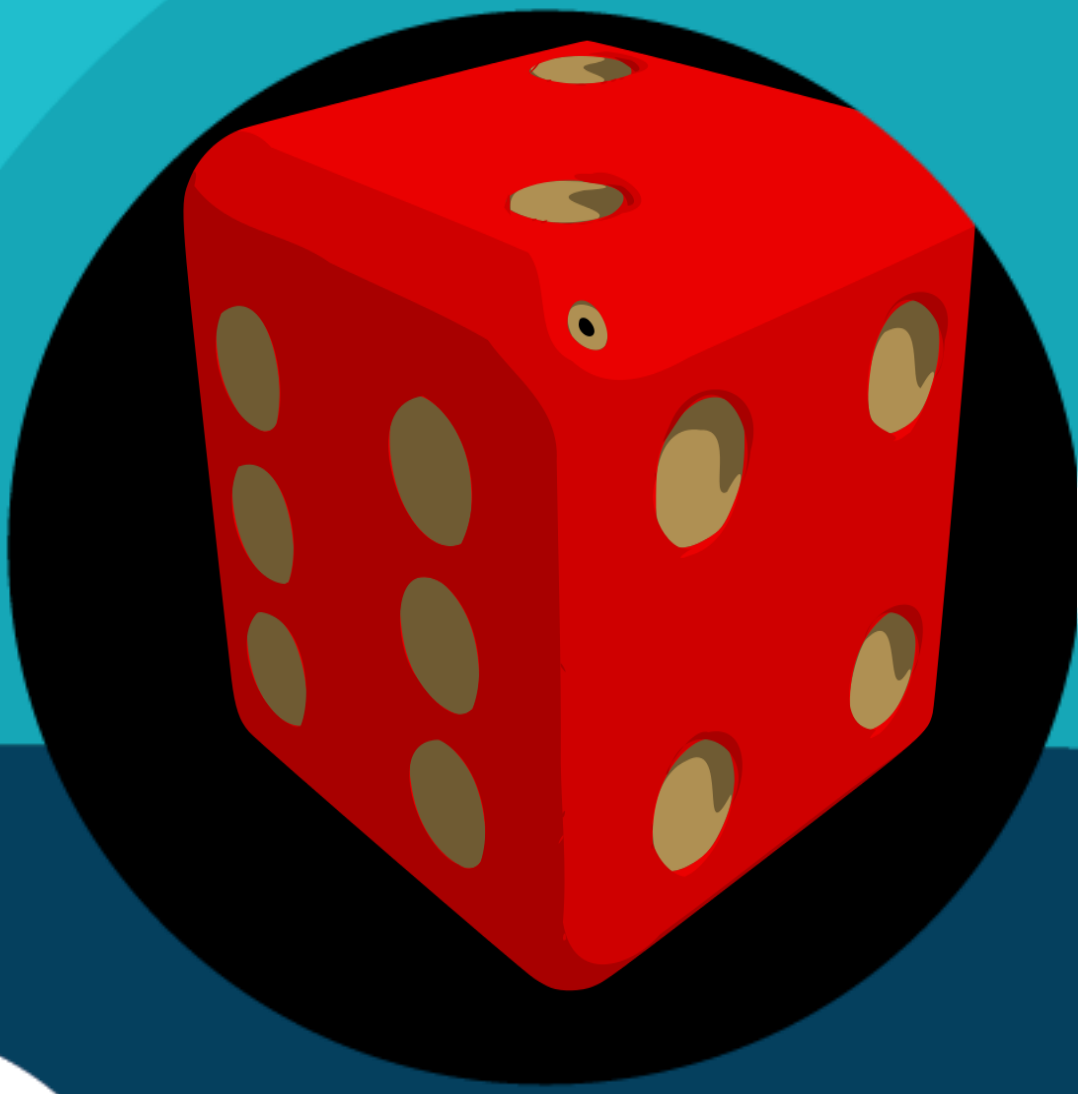
2020



GENERALA RPC

Cerdá Gianni Lucas

Garat Manuel

Sistemas Distribuidos

Departamento de Ciencias e Ingeniería de la Computación

GENERALA

Enunciado

Este proyecto consiste en diseñar e implementar el juego de la Generala por menos para 2 jugadores ubicados en diferentes máquinas. Cuando un jugador está en su turno el/los otro/s jugador/es no pueden realizar ninguna actividad en el juego. Cada jugador debe poder visualizar los dados en la pantalla al mismo tiempo.

Consideraciones:

1. Cualquier jugador puede realizar la invitación para comenzar el juego.
2. Antes de iniciar el juego se deben poner de acuerdo cuál es el orden de los jugadores.

Actividades

1. Elegir uno de los problemas presentados.
2. Capa de interfaz (presentación): sirve para facilitar la comunicación con el usuario, básicamente se considera el conjunto de pantalla para ingresar los datos y la selección de las funciones.
3. Capa de diseño e implementación del problema: Especificar el modelo diseñado para resolver el problema. Dónde se ubican cada uno de los módulos, la forma de comunicación.
4. Realizar la implementación del mismo

Descripción del Proyecto

Fuentes:

conexion.c
conexion.h
constantes.h
generala.x
gui.c
gui.h
logica.c
logica.h
main.c
servidor.c

Compilación:

make *(previamente deber instalar los* paquetes *necesarios para visualizar la GUI de la interfaz*)

- A través del siguiente comando se instalan los paquetes requeridos:
  sudo apt install build-essential libgtk-3-dev

Otros:

Generala.glade → contiene la información de la GUI en un formato XML
Fotos/* → contiene las fotos de los dados de la GUI

Ejecución: Se requiere que tres procesos corran a la vez

./servidor

./generala <IP del Servidor> (Jugador 1)

./generala <IP del Servidor> (Otro Jugador)

• Modo de uso:

Una vez que se generaron los archivos a través del makefile, se ejecuta el archivo servidor y luego los distintos clientes. Cada cliente recibe como parámetro la dirección ip del servidor, ya que debe comunicarse con él.
Una vez ejecutado el primer cliente, el mismo esperará por el registro en el server de otro cliente o jugador.
Una vez registrado el segundo cliente, el primer jugador dará comienzo a la partida.

• Comunicación

La lógica del juego se simuló utilizando un único servidor centralizado y dos clientes.
Los mismos se comunican y sincronizan sus turnos mediante el modelo de comunicación RPC.

• Módulos

Los módulos se dividen en:

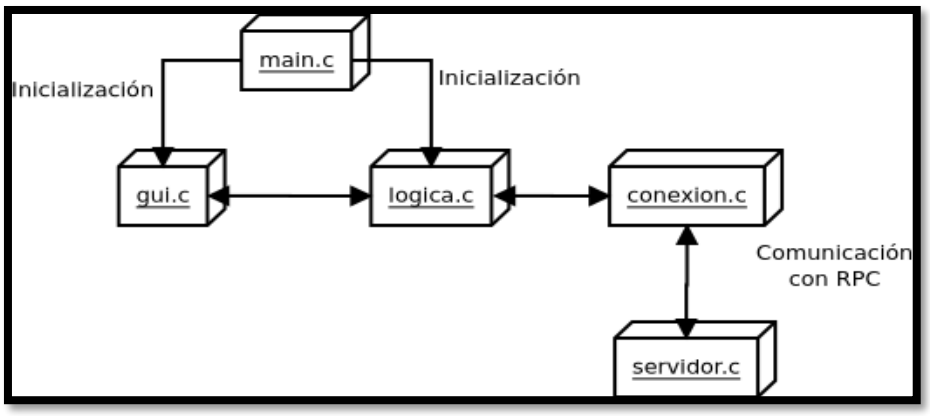
• ***gui.c***: maneja la interfaz gráfica del sistema, desarrollada con la herramienta Glade, para construirla de manera sencilla e importarla desde el módulo en un formato XML, y la librería GTK. Este módulo notifica a *logica.c* (definido más adelante) de lo que hace el usuario, y a su vez, espera acciones de este para determinar qué hacer a continuación.

• ***logica.c***: se encarga de, como bien dice su nombre, la lógica del juego: a partir de lo que notifica la interfaz gráfica, determina qué se debe hacer a continuación, en función del estado del juego. Esto puede ser: anotar o tachar una categoría, determinar qué categoría logró a partir de los dados que obtuvo, etc. También se encarga de notificar a *conexion.c* (definido más adelante) de cuándo finalizó su turno, y a su vez, espera a que este le avise de cuándo empezar el próximo turno.

• ***servidor.c***: maneja las RPC de los jugadores para iniciar y terminar el juego, determinar a quién le toca jugar, mantener la consistencia, etc.

• ***conexion.c***: se encarga de enviar las RPC al servidor cuando *logica.c* se lo indica.

• ***main.c***: simplemente inicializa los módulos de *logica.c* y *gui.c*.



- # Server

El servidor deberá mantener el estado y la información de cada jugador, por un lado, la tabla en el que se describe las categorías con el puntaje asociado a cada una de ellas como la sincronización de los turnos de cada jugada de los integrantes del juego.

El servidor le asigna y mantiene un identificador por cada participante, este identificador se otorga en el momento que el jugador inicia el juego. No se puede dar comienzo al juego hasta que haya al menos dos participantes, por lo tanto, se le informa al jugador que debe esperar por la llegada de nuevos jugadores.
Cuando un participante se anota para jugar, se le informa que debe esperar por otros jugadores. Una vez iniciado el juego, no podrán anotarse más jugadores y se cierra la cantidad de jugadores totales en la partida.

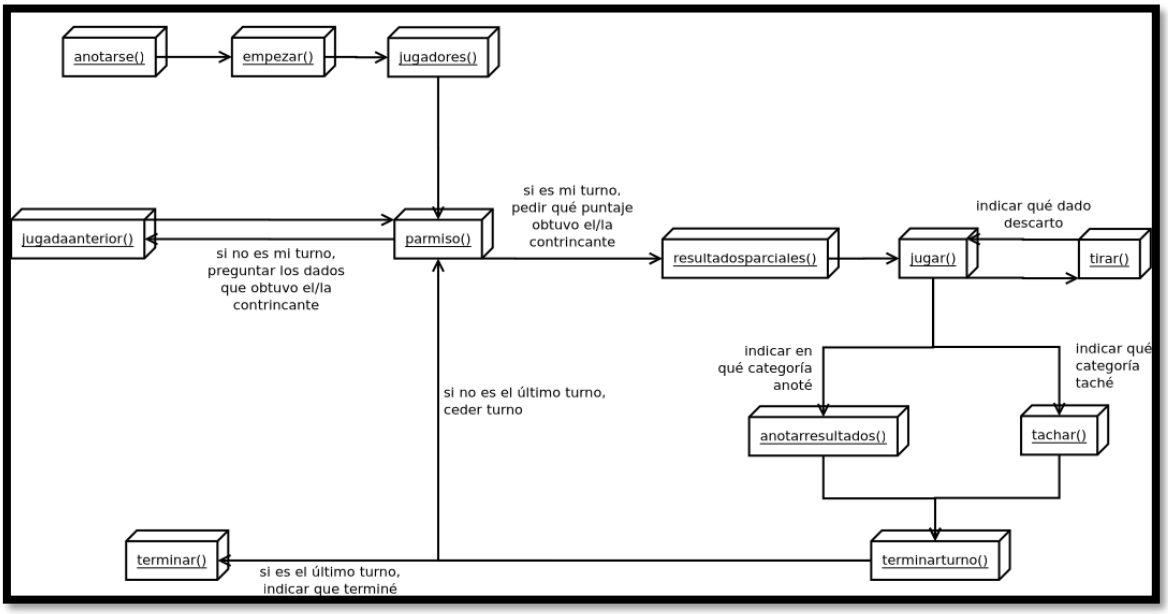
En el instante que el jugador obtiene el turno, el servidor provee las algunas de las siguientes operaciones:

- **anotarresultados()** : Se anota los puntos que el jugador actual obtuvo en sus tres tiros , estos puntos se agregan a la tabla asociada a dicho jugador.

- **jugar()**: simula la tirada de los 5 dados por parte del jugador, estos valores son aleatorios tomando dichos valores entre 1 y 6 el cual corresponde a cada número del dado.

- **tachar():**anula en la tabla del jugador actual la categoría que éste quiera tachar. A la categoría correspondiente le asigna el valor cero.

Se agrega el siguiente diagrama de flujo con el orden en que el servidor ejecuta las funciones:



Orden de los Turnos

El orden establecido de los turnos, es el orden en que los jugadores se anotaron en el juego. Cada jugador pide permiso al servidor para jugar hasta que el servidor se lo otorgue (provisionalmente, si no es su turno, los demás jugarán hasta que sea su turno).

El servidor eventualmente le otorgará el permiso, momento en el cuál registra quién es el jugador que está en turno, y le cede el permiso exclusivo a él.

Finalización del Turno y Siguiente Jugador

Una vez finalizada la jugada, el participante le indica al servidor, mediante la invocación de la función **permiso(),** que ha finalizado su turno y que puede continuar. Sólo aquí el servidor actualiza el siguiente jugador que puede jugar y cuando éste lo invoque podrá darle el permiso.

• Cliente o Jugador

El cliente en este modelo de comunicación, es el representante del jugador. Permite la interacción con el mismo, se le ofrece la interfaz correspondiente para llevar a cabo la jugada, y solicita al servidor las funciones anteriormente mencionadas.
Lleva el control de los valores anotados y tachados para no anotar o tachar dos veces en la misma categoría. Además, cuando el cliente solicita anotar una jugada, corrobora que se haya logrado una categoría válida (Desde los números 1 a 6, Escalera, Full, Póker, y demás categorías).
Una vez que el cliente se anota en el juego, el servidor lo registra, asignándole un identificador, y este espera por otros participantes, una vez que se haya registrado otro jugador, el primer cliente inicia la partida del juego.

El cliente podrá ver en su turno la tabla de su adversario para anotar o tachar los valores que obtenga, esto es importante a la hora de tomar una decisión estratégica con respecto a que categoría deberá anotar o tachar, esta información puede ser decisiva en la victoria del juego.

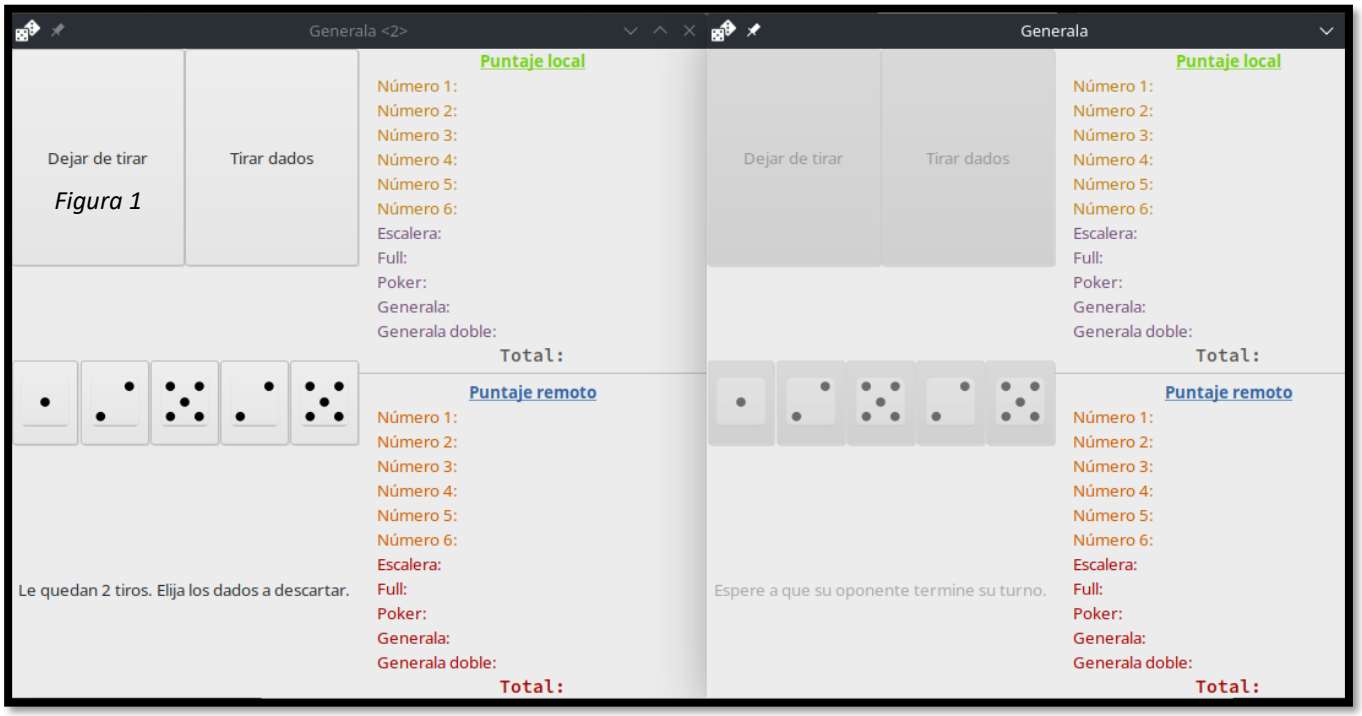


Figura 1

Se realiza un ciclo constante en el juego donde cada jugador posee 11 turnos con un máximo de 3 tiradas por turno, el cliente siempre le pide permiso al servidor para jugar. Una vez que cada participante termina sus 11 permisos, se realiza un conteo del puntaje total que obtuvo cada uno para ver quién es el ganador.
Durante cada turno la interacción es la que se mencionó en el caso del servidor, eligiéndose las operaciones a ejecutar en función de las decisiones del usuario.

• Limitaciones

Se describen las siguientes limitaciones de este proyecto:

- No está la posibilidad de tachar si hay categorías disponibles para anotar.

- Si el servidor falla, las peticiones de los clientes o jugadores del juego no podrán ser atendidas y deberán esperar a que el servidor se recupere, en consecuencia, se deberá reiniciar tanto el servidor como los participantes dentro del juego.

- Si hay un jugador que se ejecuta en una terminal esperando por otros jugadores y sale del registro del servidor y este mismo participante se vuelve a ejecutar sobre la misma terminal, entonces el servidor lo anotará como un nuevo jugador y dará comienzo al juego, por lo tanto, no será posible diferenciar jugadores que se ejecuten en una misma terminal como nuevos jugadores que se ejecuten en distintas terminales.