MANUAL DE USO Y ESPECIFICACIONES SEGUNDA PRÁCTICA

CRISTIAN CAMILO AGUILERA POLANCO
DAVID FERNANDO GUERRERO ÁLVAREZ

Ing. CESAR AUGUSTO PEDRAZA BONILLA
(Docente)

UNIVERSIDAD NACIONAL DE COLOMBIA
SEDE BOGOTÁ
FACULTAD DE INGENIERÍA
SISTEMAS OPERATIVOS
BOGOTÁ D.C
2017

MANUAL DE USO Y ESPECIFICACIONES

Este programa gestiona la información de mascotas que llegan a una veterinaria mediante dos programas principales, un servidor encargado de gestionar toda la información, y un cliente encargado de realizar solicitudes o peticiones a dicho servidor.

El programa cliente cuenta con un menú que le permite al usuario realizar ciertas operaciones, estas operaciones serán ejecutas por el servidor el cual le enviara una respuesta al cliente dependiendo la opción seleccionada.

Dicho menú se presenta a continuación:

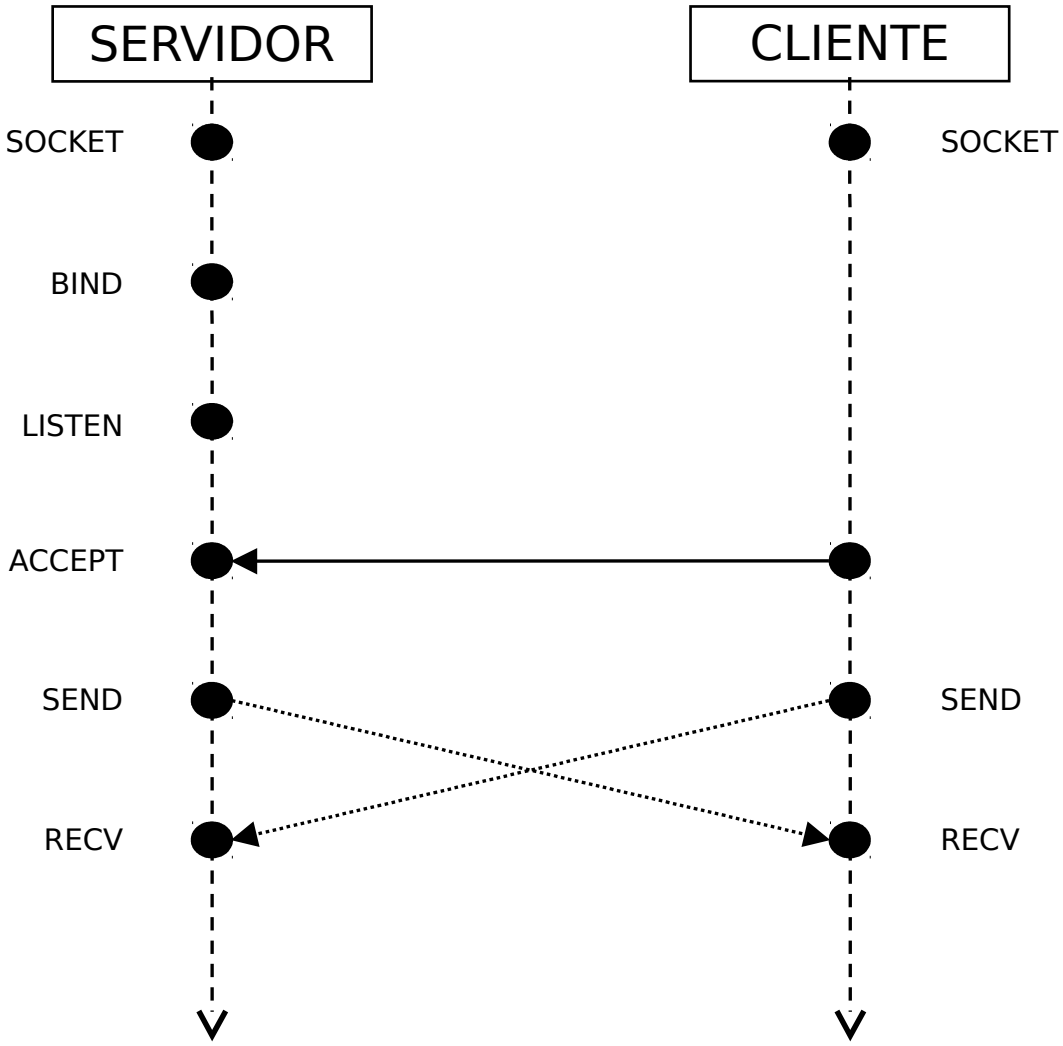
1. Ingresar Registro:
En esta opción, el cliente podrá ingresar una nueva mascota, dada la petición de ciertos datos específicos.
2. Ver Registro:
El cliente podrá visualizar el número total de mascotas registradas para luego ver en un archivo los datos completos de una mascota en específico; esta búsqueda se hará por medio de un número de registro de dicha mascota.
3. Borrar Registro:
En esta opción, el cliente verá nuevamente la cantidad total de mascotas para luego borrar un registro específico del archivo de datos de la veterinaria.
4. Buscar Registro:
Aquí, el cliente podrá realizar una búsqueda general de los registros con el mismo nombre solicitado, obteniendo así la información de todas y cada una de las mascotas que coinciden con la búsqueda.
5. Salir: Cierra la comunicación con el servidor.

Por cada opción ejecutada, siempre se da un mensaje de confirmación y se solicita presionar la tecla “enter” para continuar, antes de volver al menú principal donde podrá volver a realizar una solicitud nueva. Cuando se digita una opción, esta se envía al servidor a fin de iniciar su gestión.

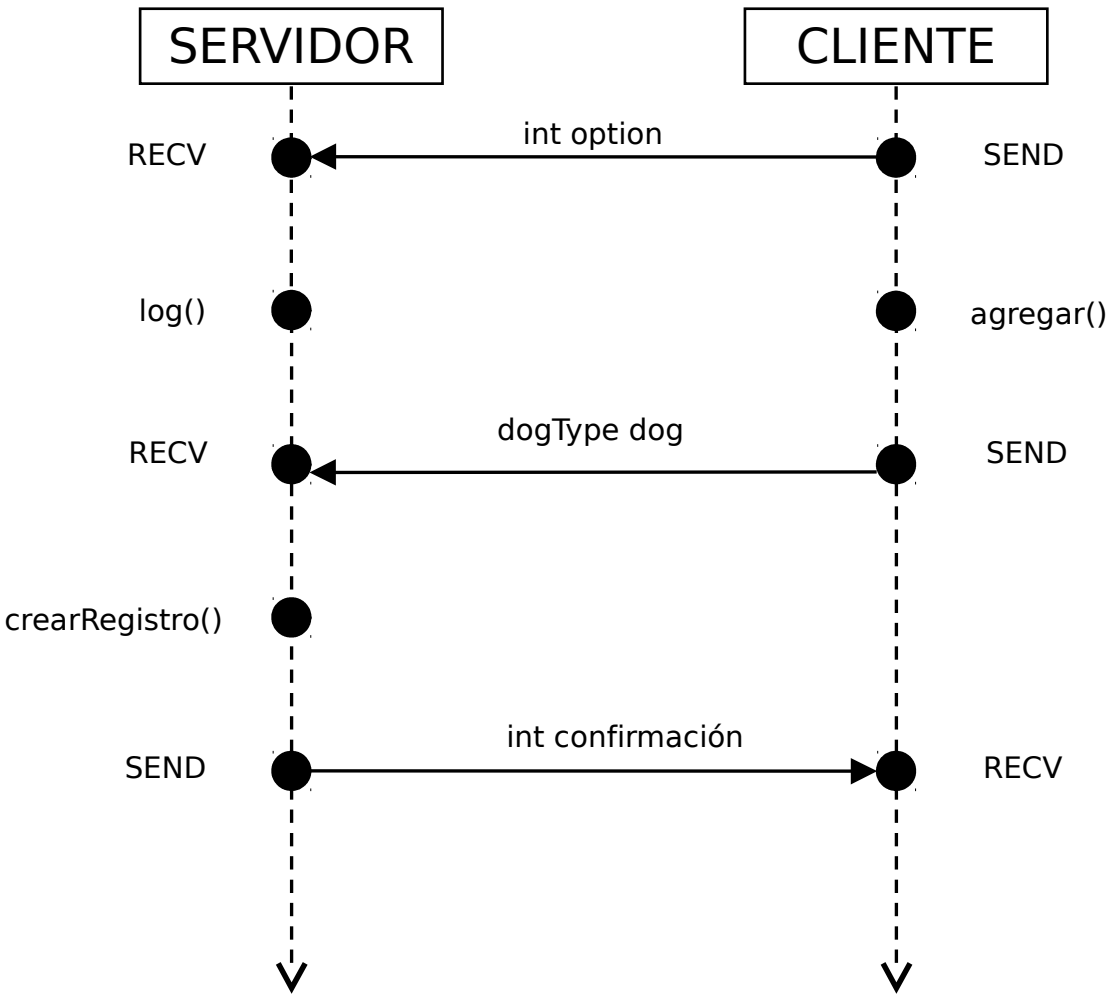
El programa servidor, gestiona todas las solicitudes del cliente, haciendo base en el archivo dataDogs.dat, que almacena los registros de las mascotas y en una tabla hash encargada de la búsqueda de los registro tras una solicitud por nombre.
Dicho servidor puede soportar 32 clientes que pueden realizar solicitudes simultaneas dado el menú presentado anteriormente.
El servidor guarda en el archivo serverDogs.log los datos del cliente y las operaciones hechas por el mismo en el siguiente formato: [Fecha YYYYMMDDTHHMMSS] Cliente [IP] [inserción | lectura | borrado | búsqueda] [registro| cadena buscada].

DIAGRAMA DE COMUNICACIONES

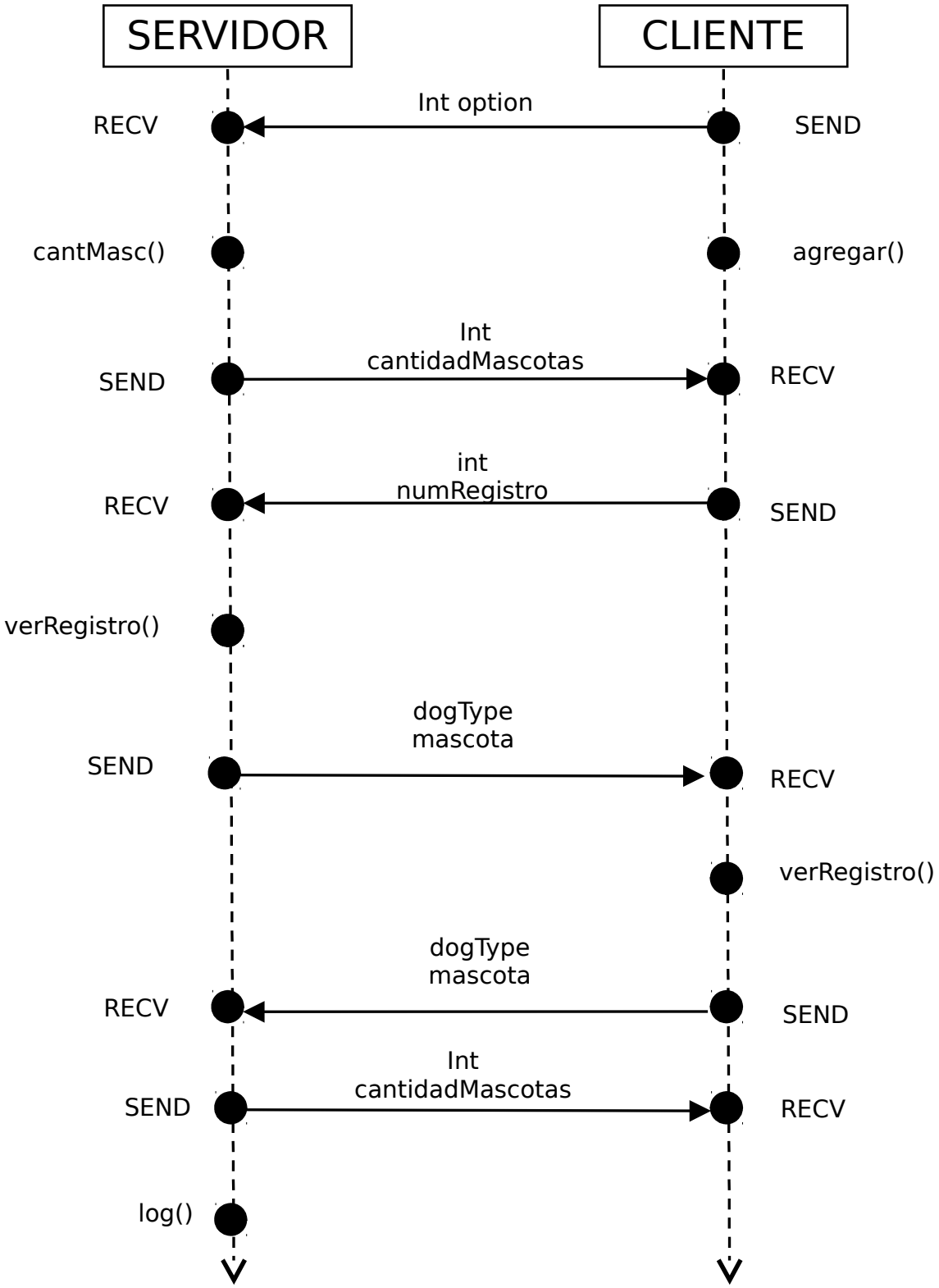
SERVIDOR – CLIENTE … CONEXIÓN



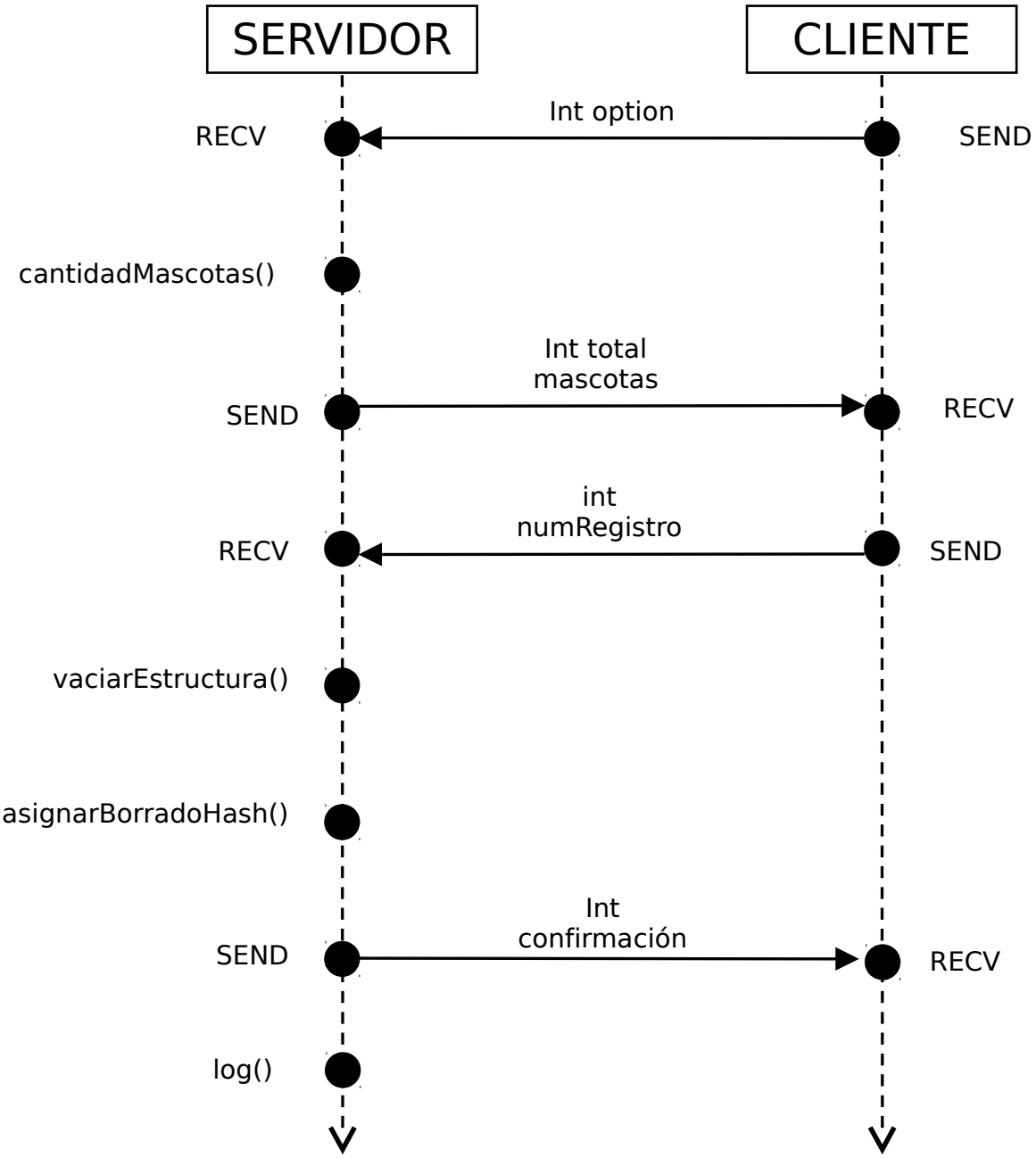
SERVIDOR – CLIENTE … INGRESAR



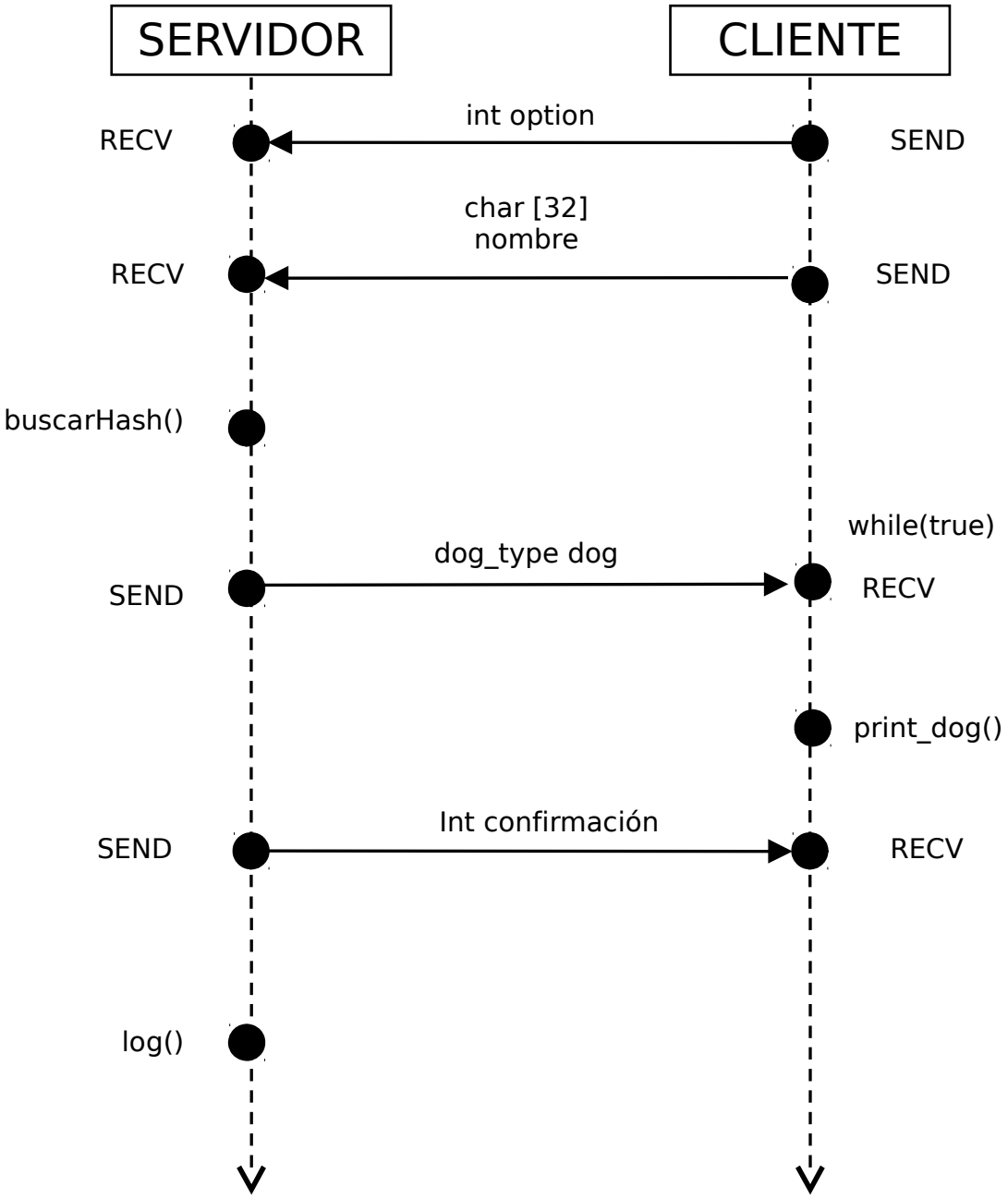
SERVIDOR – CLIENTE … VER REGISTRO



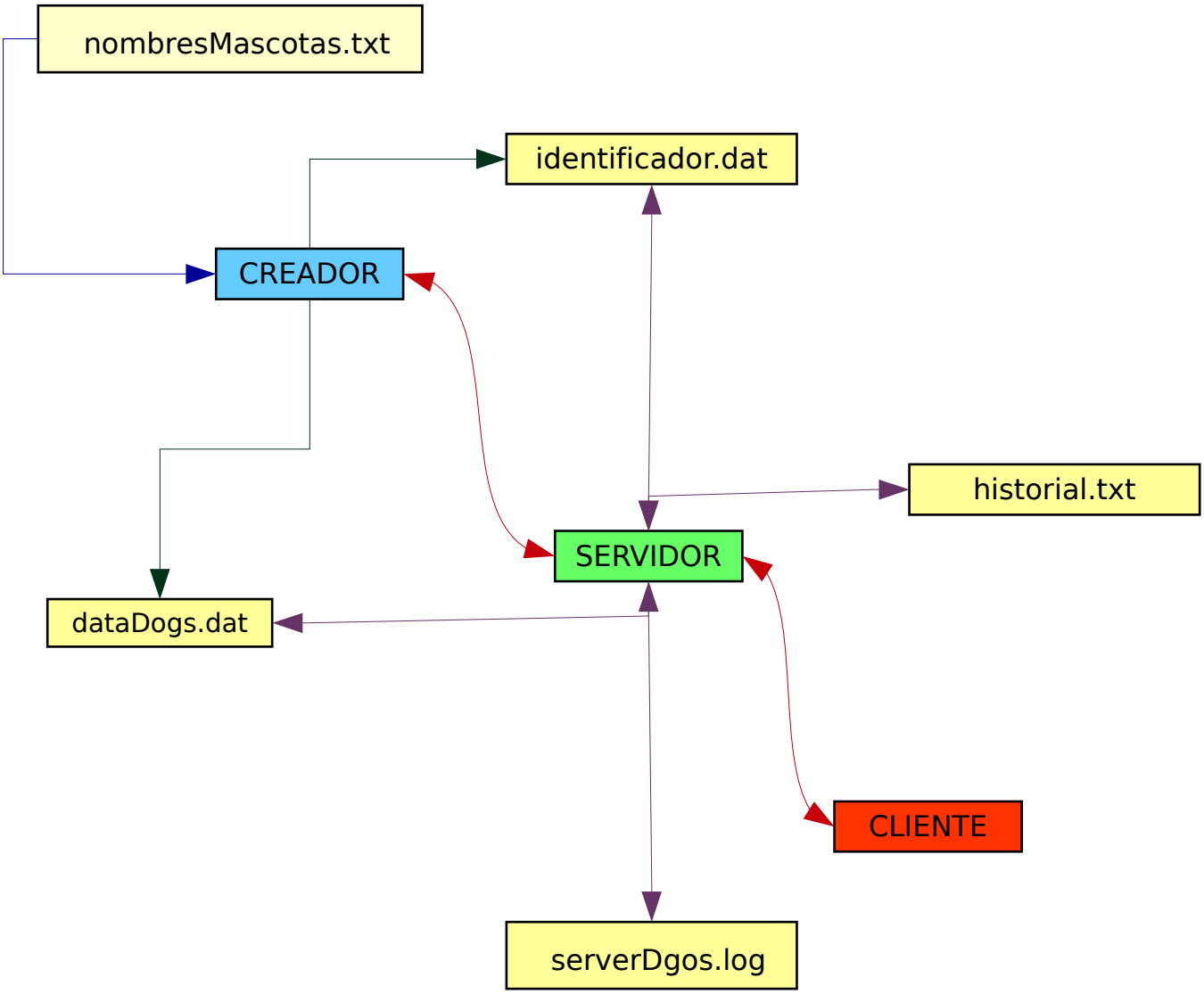
SERVIDOR – CLIENTE … BORRAR

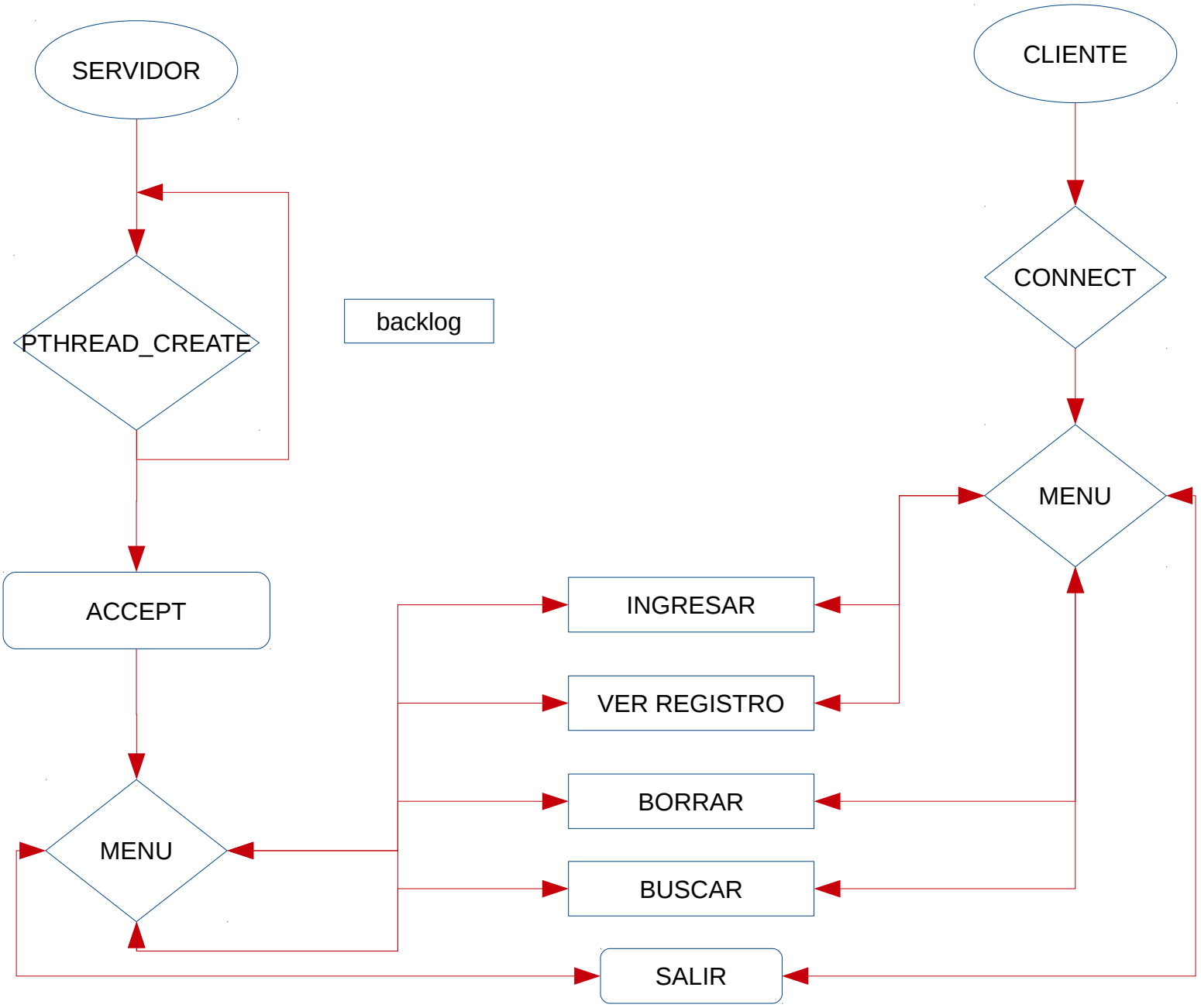


SERVIDOR – CLIENTE … BUSCAR



DIAGRAMAS DE BLOQUE




SERVIDOR
CLIENTE
PTHREAD_CREATE
backlog
CONNECT
MENU
ACCEPT
INGRESAR
VER REGISTRO
BORRAR
BUSCAR
MENU
SALIR

INFORME ELABORACIÓN – DESCRIPCIÓN

SERVIDOR

Estructuras:

- **dogType:** Contiene todos los datos de una mascota.
- **identificador:** Contiene tiene la ubicación de las mascotas en el archivo.

Funciones:

- **long cantidadMascotasArchivo(char archivo[32]):**
  Retorna la cantidad de estructuras de tipo dogType que se encuentran en el archivo dataDogs.dat.
- **long cantidadMascotasArchivo2(char archivo[32]):**
  Retorna la cantidad de estructuras de tipo identificador en el archivo identificador.dat.
- **int getHash(char nombre[32]):** Retorna el hash del nombre de la mascota.
- **int ingresarHash(struct identificador id):**
  Ingresa una estructura dogType a la tabla hash. Retorna la posición en la hash.
- **void buscarHash(char nombre[32], int clientfd):**
  Busca todas las mascotas que tiene el nombre enviado en los parametros para enviarlas al cliente.
- **void ingresarEnArchivo(char ingresarEnArchivo[32], void *mascota):**
  Ingresa una estructura del tipo dogType en el archivo dataDogType.dat.
- **void crearRegistro(void *x):**
  Ingresa una estructura dogType en el archivo dataDogtype.data y su correspondiente estructura identificador en la tabla hash.
- **void verRegistro(int clientfd, long numeroRegistro):**
  Le envia una estructura dogType al cliente, despues de buscarla con el numero que identifica la posicion en el archivo dataDogtype.dat.
- **struct dogType vaciarEstructura(long numeroRegistro):** Hace cero la estructura especificada con la posición en el archivo dataDogType.dat enviada en los parámetros. Retorna la estructura a llenar con ceros.
- **void asignarBorradoHash(struct dogType mascota, long numeroRegistro):**
  El miembro borrado de la estructura Nodo es cambiado por la letra b para identificar que identificadores de la hash deben ser borrados.
- **void actualizar():**
  Eliminar las estructuras dogType del archivo dataDogType.dat.
- **void menu(int clientfd, struct sockaddr_in client, char ip[32]):**
  EJecuta las opciones selecionadas por el cliente: Ingresar mascota, ver mascota, eliminar mascota, buscar mascota y salir de la aplicación.
- **void recibirCliente():**
  Recibe cada unno de los clientes que solicitan una conexión. Esto, mediante el accept.
- **void prepararServer():**
  Configura el servidor para poder recibir cada uno de los clientes.

CLIENTE

Estructuras:

- **dogType:** Contiene todos los datos de una mascota.

Funcione:

- **void configurar() :**
  Configura el cliente para realizar la conexión con el servidor, finalmente solicitada por connect.
- **struct dogType *agregar():**
  Retorna una estructura dogType apartir del ingreso de los datos por consola.
- **struct dogType verRegistro(struct dogType auxiliar):**
  Retorna una estructura dogType modificada luego de recibir una por parámetros y abrir la en un editor de texto y modificarla.
- **main():**
  Principalmente maneja el menú principal haciendo llamado a las funciones necesarias para cada solicitud.