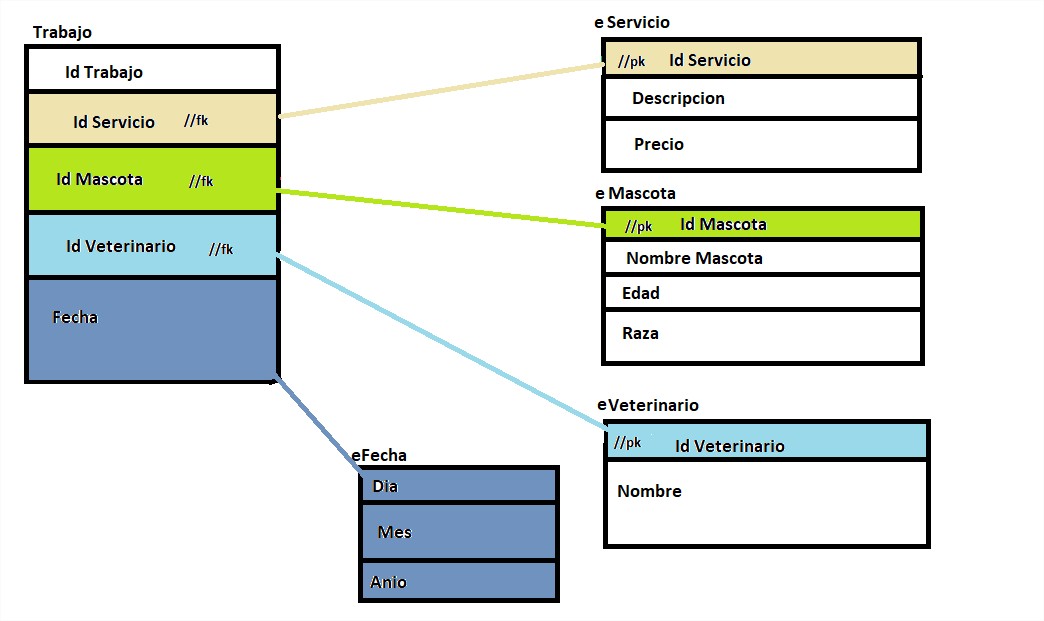
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Universidad Tecnológica Nacional Facultad Regional Avellaneda** | | | | | | | | | | |
| Técnico Universitario en Programación  Técnico Universitario en Sistemas Informáticos | | | | | | | | | | |
| Materia:    Laboratorio de computación I | | | | | | | | | | |
| Apellido: | Nieva | | | | Fecha: | |  | | | |
| Nombre: | Julian Leandro | | | | Docente: | |  | | | |
| División: | 1°C | | | | Nota: | |  | | | |
| Legajo: | 111110 | | | | Firma: | |  | | | |
| Instancia | PP |  | RPP | X | SP |  | RSP |  | FIN |  |

**Diagrama Entidad-Relación**

****

**Enunciado de los informes anexados**

***Listar trabajos que tenga un servicio y mascota especifica*: Le pregunto al usuario que servicio y mascota desea filtrar los trabajos**

***Listar veterinarios con mas trabajos en un servicio específico:* Le pido al usuario que ingrese un servicio para contar a los veterinarios que trabajen en dicho servicio**

**Prototipo de funciones**

**Trabajos.h**

/\*\*

\* @fn void InicializarTrabajos(eTrabajo[])

\* @brief Incizalizo los trabajos para que esten vacios

\*

\* @param listaTrabajos[] Array de trabajos

\*/

**void** InicializarTrabajos(eTrabajo[],**int**);

/\*\*

\* @fn int BuscarLibre(eTrabajo[])

\* @brief Buusco un espacio libre en el array

\*

\* @param listaTrabajos[] Array de trabajos

\* @return Retorna la posicion donde se encuetra libre

\*/

**int** BuscarLibre(eTrabajo[],**int**);

/\*\*

\* @fn int DarDeBajaDeTrabajo(eTrabajo[])

\* @brief Dar de baja un trabajo

\*

\* @param listadoTrabajos[] array de trabajos

\* @return Retorna un valor dependiendo de lo que sucedio dentro de la funcion

\*/

**int** DarDeBajaDeTrabajo(eTrabajo[],**int**);

/\*\*

\* @fn int BuscarId(eTrabajo[], int)

\* @brief Busca el id de un trabajo

\*

\* @param listadoTrabajos[] Array de trabajos

\* @param id Id que se le pasa mediante parametros para realizar la busqueda

\* @param tam Tamaño del array

\* @return Devuelve la posicion del array donde se encuentra el trabajo

\*/

**int** BuscarId(eTrabajo[], **int**,**int**);

**Servicio.h**

/\*\*

\* @fn void IncializarServicios(eServicio[], int)

\* @brief Incializo el array de servicio para que esten vacios

\*

\* @param servicios[] Array de servicios

\* @param tam Tamaño del array de servicios

\*/

**void** IncializarServicios(eServicio[],**int**);

/\*\*

\* @fn void HardcodearServicios(eServicio[])

\* @brief Hardcodeo el listado de los servicios

\*

\* @param listaServicios[] Array que se le insertaran los datos

\* @param tam Tamaño del array de servicios

\*/

**void** HardcodearServicios(eServicio[],**int**);

/\*\*

\* @fn int ValidarIdServicio(char[], int, eServicio[], int)

\* @brief Valido el id del servicio que ingreso el usuario

\*

\* @param mensajeError[] Mensaje de error que se mostrara si es invalido el id ingresado

\* @param numeroAValidar Es el id que sera validado

\* @param servicio[] Array de servicios

\* @param tam Tamaño del array para la iteracion

\* @return Retorna El id validado

\*/

**int** ValidarIdServicio(**char**[], **int**, eServicio[], **int**);

/\*\*

\* @fn void MostrarServicios(eServicio[])

\* @brief Muestra los servicios

\*

\* @param servicio[] Array de servicios que seran mostrados

\*/

**void** MostrarServicios(eServicio[],**int**);

/\*\*

\* @fn void MostrarUnServicio(eServicio)

\* @brief Muestra un solo servicio

\*

\* @param servicio Servicio recibido y que sera mostrado

\*/

**void** MostrarUnServicio(eServicio,**int**);

/\*\*

\* @fn eServicio DameUnServicio(eServicio[], int)

\* @brief Me da un servicio con el id recibido

\*

\* @param servicios[] Array de servicios

\* @param id Id del trabajo

\* @return Retorna un tipo de dato eServicio

\*/

eServicio DameUnServicio(eServicio[], **int**,**int**);

**Mascota.h**

/\*\*

\* @fn void InicializarMascotas(eMascota[], int)

\* @brief Inicializo el array de mascotas para que esten vacios

\*

\* @param mascotas[] Array de mascotas

\* @param tam Tamaño del array de mascotas

\*/

**void** InicializarMascotas(eMascota[],**int**);

/\*\*

\* @fn void HardcodearMascotas(eMascota[], int)

\* @brief Hardcodeo el listado de mascotas

\*

\* @param mascotas[] Array de mascotas

\* @param tam Tamaño del array de mascotas

\*/

**void** HardcodearMascotas(eMascota[],**int**);

/\*\*

\* @fn void MostrarMascotas(eMascota[], int)

\* @brief Muestro las mascotas cargadas en la lista

\*

\* @param mascotas[] Array de mascotas

\* @param tam Tamaño del array de mascotas

\*/

**void** MostrarMascotas(eMascota[],**int**);

/\*\*

\* @fn int ValidarIdMascota(char[], int, eMascota[], int)

\* @brief Valido el id de la mascota que ingreso el usuario

\*

\* @param mensajeError[] Mensaje de error

\* @param numeroAValidar Id a ser validad

\* @param mascotas[] Array de mascotas

\* @param tam Tamaño del array de mascotas

\* @return Retorna El id validado

\*/

**int** ValidarIdMascota(**char**[],**int**, eMascota[],**int**);

/\*\*

\* @fn eMascota DameUnaMascota(eMascota[], int, int)

\* @brief Obtengo una mascota con el id recibido y la devuelvo

\*

\* @param mascotas[] Array de mascotas

\* @param id ID de la mascota a buscar

\* @param tam Tamaño del array de mascotas

\* @return Devuelvo la mascota encontrada

\*/

eMascota DameUnaMascota(eMascota[],**int**,**int**);

**Veterinario.h**

/\*\*

\* @fn void InicializarVeterinarios(eVeterinario[], int)

\* @brief Inicializo a los veterinarios en VACIO

\*

\* @param listaVeterinarios Array de veterinarios

\* @param tam Tamaño del array de veterinario

\*/

**void** InicializarVeterinarios(eVeterinario listaVeterinarios[],**int** tam);

/\*\*

\* @fn void HardcodearVeterinarios(eVeterinario[], int)

\* @brief Hardcodeo a los veterinarios

\*

\* @param listaVeterinarios Array de veterinarios

\* @param tam Tamaño del array de veterinario

\*/

**void** HardcodearVeterinarios(eVeterinario listaVeterinarios[],**int** tam);

/\*\*

\* @fn void MostrarVeterinarios(eVeterinario[], int)

\* @brief Muestro a los veterinarios cargados actualmente

\*

\* @param listaVeterinarios Array de veterinarios

\* @param tam Tamaño del array de veterinario

\*/

**void** MostrarVeterinarios(eVeterinario listaVeterinarios[],**int** tam);

/\*\*

\* @fn void MostrarUnVeterinario(eVeterinario)

\* @brief Muestro un solo veterinario

\*

\* @param veterinario Veterinario a mostrar

\*/

**void** MostrarUnVeterinario(eVeterinario veterinario);

/\*\*

\* @fn int ValidarIdVeterinario(char[], int, eVeterinario[], int)

\* @brief Valido el id de veterinario que ingreso el usuario

\*

\* @param mensajeError Mensaje de error

\* @param numeroAValidar Id que ingreso el usuario

\* @param listaVeterinarios Array de veterinarios

\* @param tam Tamaño del array de veterinario

\* @return Devuelvo el Id del veterinario validado que ingreso el usuario

\*/

**int** ValidarIdVeterinario(**char** mensajeError[],**int** numeroAValidar,eVeterinario listaVeterinario[], **int** tam);

/\*\*

\* @fn eVeterinario DameUnVeterinario(eVeterinario[], int, int)

\* @brief Obtengo un veterinario con el id pasado como parametro

\*

\* @param listaVeterinarios Array de veterinarios

\* @param tam Tamaño del array de veterinario

\* @param id Id a buscar

\* @return Devuelvo el veterinario encontrado

\*/

eVeterinario DameUnVeterinario(eVeterinario listaVeterinarios[],**int** tam,**int** id);

**Fecha.h**

/\*\*

\* @fn eFecha PedirFecha(void)

\* @brief Pido los datos de la fecha

\*

\* @return Devuelvo la estructura a la que le cargue los datos

\*/

eFecha PedirFecha(**void**);

**Menu.h**

/\*\*

\* @fn int MenuPrincipal(void)

\* @brief Muestro el menu principal al usuario

\*

\* @return Devuelvo la opcion elegida por el usuario

\*/

**int** MenuPrincipal(**void**);

/\*\*

\* @fn int SubMenuModificar(void)

\* @brief Muestro el menu de modificar al usuario

\*

\* @return Devuelvo la opcion elegida por el usuario

\*/

**int** SubMenuModificar(**void**);

**Funciones.h**

/\*\*

\* @fn int DarAltaTrabajo(eTrabajo[], int, eServicio[], int, eMascota[], int, eVeterinario[], int, int\*)

\* @brief Cargo un trabajo a la lista

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño del array de trabajos

\* @param servicios Array de servicios

\* @param tamServicios Tamaño del array de servicios

\* @param mascotas Array de mascotas

\* @param tamMascotas Tamaño del array de mascotas

\* @param veterinarios Array de veterinarios

\* @param tamVeterinarios Tamaño del array de veterinarios

\* @param idTrabajo Puntero al id que se le asignara al nuevo trabajo

\* @return Devuelvo -1 si no hay mas espacio, Devuelvo un numero distinto a -1 si se agrego a la lista

\*/

**int** DarAltaTrabajo(eTrabajo listaTrabajos[], **int** tamTrabajos,eServicio servicios[],**int** tamServicios,eMascota mascotas[],**int** tamMascotas,eVeterinario veterinarios[],**int** tamVeterinarios, **int**\* idTrabajo);

/\*\*

\* @fn void MostrarTrabajos(eTrabajo[], int, eServicio[], int, eMascota[], int, eVeterinario[], int)

\* @brief Muestro la lista de trabajos cargados

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño del array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño del array de servicios

\* @param listaMascotas Array de mascotas

\* @param tamMascotas Tamaño del array de mascotas

\* @param listaVeterinarios Array de veterinarios

\* @param tamVeterinario Tamaño del array de veterinarios

\*/

**void** MostrarTrabajos(eTrabajo listaTrabajos[],**int** tamTrabajos, eServicio listaServicios[],**int** tamServicios, eMascota listaMascotas[],**int** tamMascotas,eVeterinario listaVeterinarios[],**int** tamVeterinario);

/\*\*

\* @fn int InicializarModificacionDeTrabajo(eTrabajo[], int, eServicio[], int, eMascota[], int, eVeterinario[], int)

\* @brief Inicio el proceso de modificarion de un trabajo

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño del array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño del array de servicios

\* @param listaMascotas Array de mascotas

\* @param tamMascotas Tamaño del array de mascotas

\* @param listaVeterinarios Array de veterinarios

\* @param tamVeterinario Tamaño del array de veterinarios

\* @return Devuelvo -1 si no se encontro el trabajo, 0 si se cancelo la operacion de modificacion, 1 si acepto la modificacion

\*/

**int** InicializarModificacionDeTrabajo(eTrabajo listaTrabajos[], **int** tamTrabajos,eServicio listaServicios[],**int** tamServicios,eMascota listaMascotas[],**int** tamMascotas,eVeterinario listaVeterinarios[], **int** tamVeterinarios);

/\*\*

\* @fn void OrdenarTrabajosPorAnio(eTrabajo[], int, eMascota[], int)

\* @brief Ordeno los trabajos por año

\*

\* @param trabajos Array de trabajos

\* @param tam Tamaño del array de trabajos

\* @param mascotas Array de mascotas

\* @param tamMascotas Tamaño del array de mascotas

\*/

**void** OrdenarTrabajosPorAnio(eTrabajo trabajos[],**int** tam,eMascota mascotas[],**int** tamMascotas);

/\*\*

\* @fn void OrdenarTrabajosPorNombreMascota(eTrabajo[], int, eMascota[], int)

\* @brief Ordeno los trabajos por nombre de mascota

\*

\* @param trabajos Array de trabajos

\* @param tam Tamaño del array de trabajos

\* @param mascotas Array de mascotas

\* @param tamMascotas Tamaño del array de mascotas

\*/

**void** OrdenarTrabajosPorNombreMascota(eTrabajo trabajos[], **int** tam,eMascota mascotas[],**int** tamMascotas);

/\*\*

\* @fn void MostrarPrecioTotalPorLosServicioPrestados(eTrabajo[], int, eServicio[], int)

\* @brief Muestro el precio total de los servicios cargados en la lista de trabajos

\*

\* @param trabajos Array de trabajos

\* @param tam Tamaño del array de trabajos

\* @param servicios Array de servicios

\* @param tamServicios Tamaño del array de servicios

\*/

**void** MostrarPrecioTotalPorLosServicioPrestados(eTrabajo trabajos[],**int** tamTrabajos, eServicio servicios[], **int** tamServicios);

/\*\*

\* @fn void MostrarServiciosConMasTrabajosRealizados(eTrabajo[], int, eServicio[], int)

\* @brief Inicio el proceso de mostrar los servicios con mas trabajos

\*

\* @param trabajos Array de trabajos

\* @param tam Tamaño del array de trabajos

\* @param servicios Array de servicios

\* @param tamServicios Tamaño del array de servicios

\*/

**void** MostrarServiciosConMasTrabajosRealizados(eTrabajo trabajos[],**int** tamTrabajos, eServicio servicios[],**int** tamServicios);

/\*\*

\* @fn void ListarServiciosConMascotas(eTrabajo[], int, eServicio[], int, eMascota[], int)

\* @brief Listo los servicios con la mascota que le corresponde

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño de array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño de array de servicios

\* @param listaMascotas Array de mascotas

\* @param tamMascotas Tamaño de array de mascotas

\*/

**void** ListarServiciosConMascotas(eTrabajo listaTrabajos[],**int** tamTrabajos,eServicio listaServicios[],**int** tamServicios,eMascota listaMascotas[],**int** tamMascotas);

/\*\*

\* @fn void MostrarPromedioDeEdadesMascotasSegunServicio(eTrabajo[], int, eServicio[], int, eMascota[], int)

\* @brief Muestro el promedio de edades de mascotas segun el servicio elegido

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño de array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño de array de servicios

\* @param listaMascotas Array de mascotas

\* @param tamMascotas Tamaño de array de mascotas

\*/

**void** MostrarPromedioDeEdadesMascotasSegunServicio(eTrabajo listaTrabajos[],**int** tamTrabajos,eServicio listaServicios[],**int** tamServicios,eMascota listaMascotas[],**int** tamMascotas);

/\*\*

\* @fn void MostrarVeterinariosQueTrabajenEnUnServicioYMascotaEspecifico(eTrabajo[], int, eServicio[], int, eMascota[], int, eVeterinario[], int)

\* @brief Muestro los trabajos con un servicio y una mascota especifica ingresada por el usuario

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño de array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño de servicios

\* @param listaMascotas Array de mascotas

\* @param tamMascotas Tamaño de array de mascotas

\* @param listaVeterinarios Array de veterinarios

\* @param tamVeterinarios Tamaño del array de veterinarios

\*/

**void** MostrarTrabajosConUnServicioYMascotaEspecifica(eTrabajo listaTrabajos[],**int** tamTrabajos,eServicio listaServicios[],**int** tamServicios,eMascota listaMascotas[],**int** tamMascotas,eVeterinario listaVeterinarios[],**int** tamVeterinarios);

/\*\*

\* @fn void MostrarVeterinariosConMasTrabajosEnUnServicio(eTrabajo[], int, eServicio[], int, eVeterinario[], int)

\* @brief Muestro los veterinarios con mas trabajo en un servicio ingresado por el usuario

\*

\* @param listaTrabajos Array de trabajos

\* @param tamTrabajos Tamaño de array de trabajos

\* @param listaServicios Array de servicios

\* @param tamServicios Tamaño de servicios

\* @param listaVeterinarios Array de veterinarios

\* @param tamVeterinarios Tamaño del array de veterinarios

\*/

**void** MostrarVeterinariosConMasTrabajosEnUnServicio(eTrabajo listaTrabajos[],**int** tamTrabajos,eServicio listaServicios[],**int** tamServicios,eVeterinario listaVeterinarios[],**int** tamVeterinarios);

**Input.h**

/\*\*

\* @fn int utn\_getInt(char[], char[], int, int, int, int\*)

\* @brief Se incialializa el proceso de pedir un entero

\*

\* @param mensaje Mensaje a mostrar al usuario

\* @param mensajeError Mensaje de error por si el usuario no ingreso lo deseado

\* @param min Minimo requerido

\* @param max Maximo requerido

\* @param reintentos Cantidad de reintentos posibles para el usuario

\* @param input Variable que se desea agregar el valor ingresado por el usuario (Mediante referencia)

\* @return Devuelve -1 si hay un error (Input == NULL, mensajeError == NULL, mensaje == NULL ,min invalido, max invalido o reintentos invalido), 0 si esta bien

\*/

**int** utn\_getInt(**char** mensaje[], **char** mensajeError[], **int** min, **int** max, **int** reintentos, **int**\* input);

/\*\*

\* @fn int getInt(int\*)

\* @brief Se solicita que se escriba numeros en un string y los devuelve si se cumplen las condiciones

\*

\* @param input Puntero de la variable recibida por referencia

\* @param mensaje Mensaje a mostrar al usuario

\* @return Devuelve -1 si no se escribieron solo numeros, 0 si esta bien

\*/

**int** getInt(**int**\* input,**char** mensaje[]);

**Link al video explicativo Drive**

**https://drive.google.com/file/d/1f3mQz0aeMN-bQN1dfiS00R2R0xkLLZj7/view?usp=sharing**