Cesar Omar Algañaras – 1F  
DNI: 41823068

**Estructuras del parcial**

**typedef** **struct**

{

**int** dia;

**int** mes;

**int** anio;

}eFecha;

**typedef** **struct**

{

**int** id;

**int** idBicicleta;

**int** idFormaDePago;

**int** idServicio;

eFecha fecha;

**int** isEmpty;

}eTrabajo;  
  
**typedef** **struct**

{

**int** id;

**char** descripcion[25];

**float** precio;

**int** isEmpty;

}eServicio;

**typedef** **struct**

{

**int** idBicicleta;

**char** marca[25];

**int** rodadoBicicleta;

**char** color[25];

**int** isEmpty;

}eBicicleta;

**typedef** **struct**

{

**int** id;

**char** descripcion[TAM\_AUX];

**char** bancarizado[TAM\_AUX];

**char** numOperacion[TAM\_AUX];

**int** contador;

**float** acumulador;

**int** contadorFormaDePago;

**int** isEmpty;

}eAuxiliar;  
  
**Estructura agregada**

**typedef** **struct**

{

**int** idFormaDePago;

**char** descripcionPago[30];

**char** bancarizado[30];

**char** numOperacion[30];

}eFormaPago;

**Estructura agregada**

**typedef** **struct**

{

**int** id;

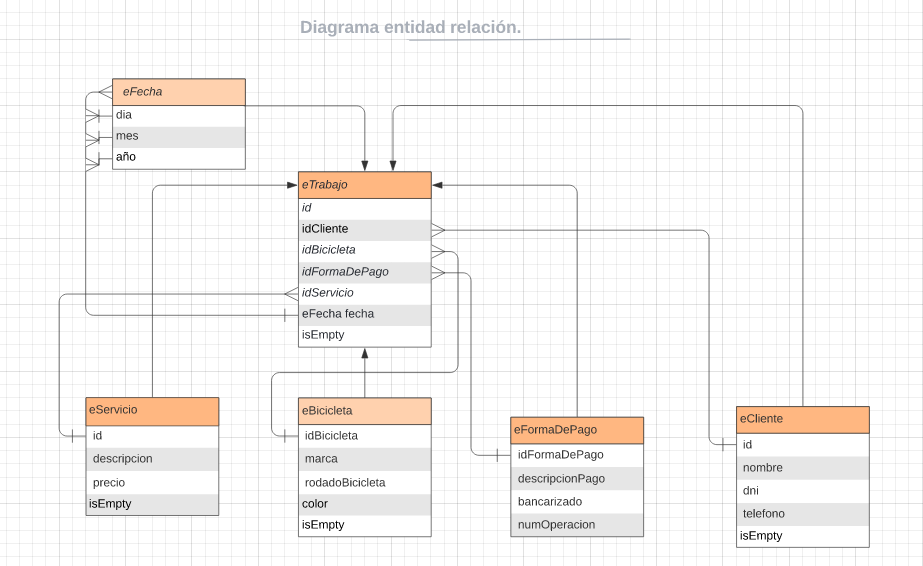
**char** nombre[50];

**int** dni;

**int** telefono;

**int** isEmpty;

}eCliente;



**ESTRUCTURA DEL MENU**

1. ALTA TRABAJO

2. MODIFICAR TRABAJO

3. BAJA TRABAJO

4. LISTADO TRABAJO

5. LISTADO SERVICIOS

6. TOTAL SERVICIOS PRESTADOS

7. LISTADO POR MARCA DE BICICLETA

8. SERVICIOS CON MAS TRABAJOS

9. LISTADO DE SERVICIOS REALIZADOS A BICICLETAS

10. CANTIDAD DE BICICLETAS ROJAS CON SERVICIO ELEGIO POR USUARIO

11. LISTADO DE TRABAJOS CON FORMAS DE PAGO (NUEVO)

12. FORMA DE PAGO MAS UTILIZADA (NUEVO)

13. LISTADO DE CLIENTES CON TRABAJOS A PAGAR (NUEVO)

**Prototipo de funciones**

1. ALTA TRABAJO

/\*\*

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

\* @brief Si hay espacio en el array pide el ingreso de datos para cargarlos

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @return 1 si se cargaron con exito los datos del trabajo o 0 si no hay espacio en el array

\*/

**int** **DarAltaTrabajo**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB, eFormaPago listaP[], **int** tamP, **int** idTrabajoIncremental);

2. MODIFICACION TRABAJO  
/\*\*

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

\* @brief Se ingresa un id donde si es valido permite modificar datos del trabajo como el servicio, la bicicleta al que se le da el servicio y cliente

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return 1 si la modificacion fue exitosa o 0 si no hay datos para modificar

\*/

**int** **ModificarTrabajo**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);  
  
3. BAJA TRABAJO  
/\*\*

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

\* @brief Si hay datos cargados te muestra un listado de los trabajos para darle de baja

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return 1 si se dio de baja un trabajo o 0 si no hay datos de trabajo cargados

\*/

**int** **DarBajaTrabajo**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);

4. LISTADO TRABAJO

/\*\*

\* @fn void OrdenarPorFecha(eTrabajo[], int, eBicicleta[], int)

\* @brief Ordena la lista de trabajos por fecha y ante igualdad de fecha por marca

\*

\* @param listaT

\* @param tamT

\* @param listaB

\* @param tamB

\*/

**void** **OrdenarPorFecha**(eTrabajo listaT[], **int** tamT, eBicicleta listaB[], **int** tamB);

/\*\*

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

\* @brief Si hay datos de trabajos cargados muestra un listado de todos ellos con su respectivo servicio y bicicleta ordena por año, y ante igualdad de año por marca

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return 1 si los datos se mostraron o 0 si no hay datos para mostrar

\*/

**int** **MostrarTrabajos**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);

5. LISTADO SERVICIOS  
/\*\*

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

\* @brief Muestra una lista de todos los servicios disponibles

\*

\* @param listaS

\* @param tamS

\*/

**int** **MostrarServicios**(eServicio listaS[], **int** tamS);  
  
6. TOTAL SERVICIOS PRESTADOS  
/\*\*

\* @fn int MostrarServiciosPrestados(eAuxiliar[], eTrabajo[], int, eServicio[], int)

\* @brief Muestra una lista del total de los servicios prestados mas el precio acumulado

\*

\* @param listaAux

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return

\*/

**int** **MostrarServiciosPrestados**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS);  
  
7. LISTADO POR MARCA DE BICICLETA

/\*\*

\* @fn void OrdenarPorMarca(eTrabajo[], int, eBicicleta[], int)

\* @brief Ordena la lista de trabajos por marca de bicicleta

\*

\* @param listaT

\* @param tamT

\* @param listaB

\* @param tamB

\*/

**void** **OrdenarPorMarca**(eTrabajo listaT[], **int** tamT, eBicicleta listaB[], **int** tamB);

/\*\*

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

\* @brief Si hay datos de trabajos cargados muestra un listado de todos ellos con su respectivo servicio y bicicleta ordena por año, y ante igualdad de año por marca

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return 1 si los datos se mostraron o 0 si no hay datos para mostrar

\*/

**int** **MostrarTrabajos**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);  
  
8. SERVICIOS CON MAS TRABAJOS

/\*\*

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

\* @brief muestra cual es el servicio con mayor cantidad de trabajos mas la cantidad, y ante igualal de trabajos muestra ambos servicios

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @return

\*/

**int** **ServicioConMasTrabajosRealizado**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS);

9. LISTADO DE SERVICIOS REALIZADOS A BICICLETAS

/\*\*

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

\* @brief Muestra una lista de bicicletas con su respectivo servicio mas el precio del mismo

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @return 1 si puede mostrar la lista 0 si no pudo

\*/

**int** **ListaBicicletasPorServicio**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);

10. CANTIDAD DE BICICLETAS ROJAS CON SERVICIO ELEGIO POR USUARIO

/\*\*

\* @fn int BicicletasRojasEligidasPorUsiario(eTrabajo[], int, eServicio[], int, eBicicleta[], int)

\* @brief Muestra la cantidad de bicicletas rojas elegidas por el usuario

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @return 1 si pudo contar una bicicleta 0 si no pudo

\*/

**int** **BicicletasRojasEligidasPorUsiario**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);

11. LISTADO DE TRABAJOS CON FORMAS DE PAGO

/\*\*

\* @fn int ListaTrabajosConFormaDePago(eTrabajo[], int, eServicio[], int, eBicicleta[], int, eFormaPago[], int)

\* @brief Muestra toda la lista de trabajos todos los datos, incluyendo servicios, bicicletaas y forma de pago

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @param listaP

\* @param tamP

\* @return 1 si puede mostrar un trabajo 0 si no pudo

\*/

**int** **ListaTrabajosConFormaDePago**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB, eFormaPago listaP[], **int** tamP);

12. LISTADO DE TRABAJOS CON FORMA DE PAGO MAS UTILIZADA

/\*\*

\* @fn int ListaFormaDePagoMasUtilizada(eTrabajo[], int, eServicio[], int, eBicicleta[], int, eFormaPago[], int)

\* @brief Muestra una lista de los trabajos pero solo con la forma de pago mas utilizada, y ante igualdad con otras formas de pago mostraria ambas

\*

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @param listaP

\* @param tamP

\* @return

\*/

**int** **ListaTrabajoFormaDePagoMasUtilizada**(eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB, eFormaPago listaP[], **int** tamP);

13. LISTADO DE CLIENTES CON TRABAJOS A PAGAR

/\*\*

\* @fn int MostrarTodosClientesConTrabajos(eCliente[], int, eTrabajo[], int, eServicio[], int, eBicicleta[], int)

\* @brief Muestra todos los clientes con la cantidad de trabajos y el precio a pagar y tiene un filtro para que el usuario elija si quiere

\* ver los servicios del cliente con mas trabajos o filtrarlo por ID

\*

\* @param listaC

\* @param tamC

\* @param listaT

\* @param tamT

\* @param listaS

\* @param tamS

\* @param listaB

\* @param tamB

\* @return 1 si pudo mostrar un cliente 0 si no pudo

\*/

**int** MostrarTodosClientesConTrabajos(eCliente listaC[], **int** tamC, eTrabajo listaT[], **int** tamT, eServicio listaS[], **int** tamS, eBicicleta listaB[], **int** tamB);

LINK DEL VIDEO  
  
https://drive.google.com/file/d/1VzK4XvnNXtzd0gw0cPtmJ2LkNIs0Di9R/view?usp=sharing