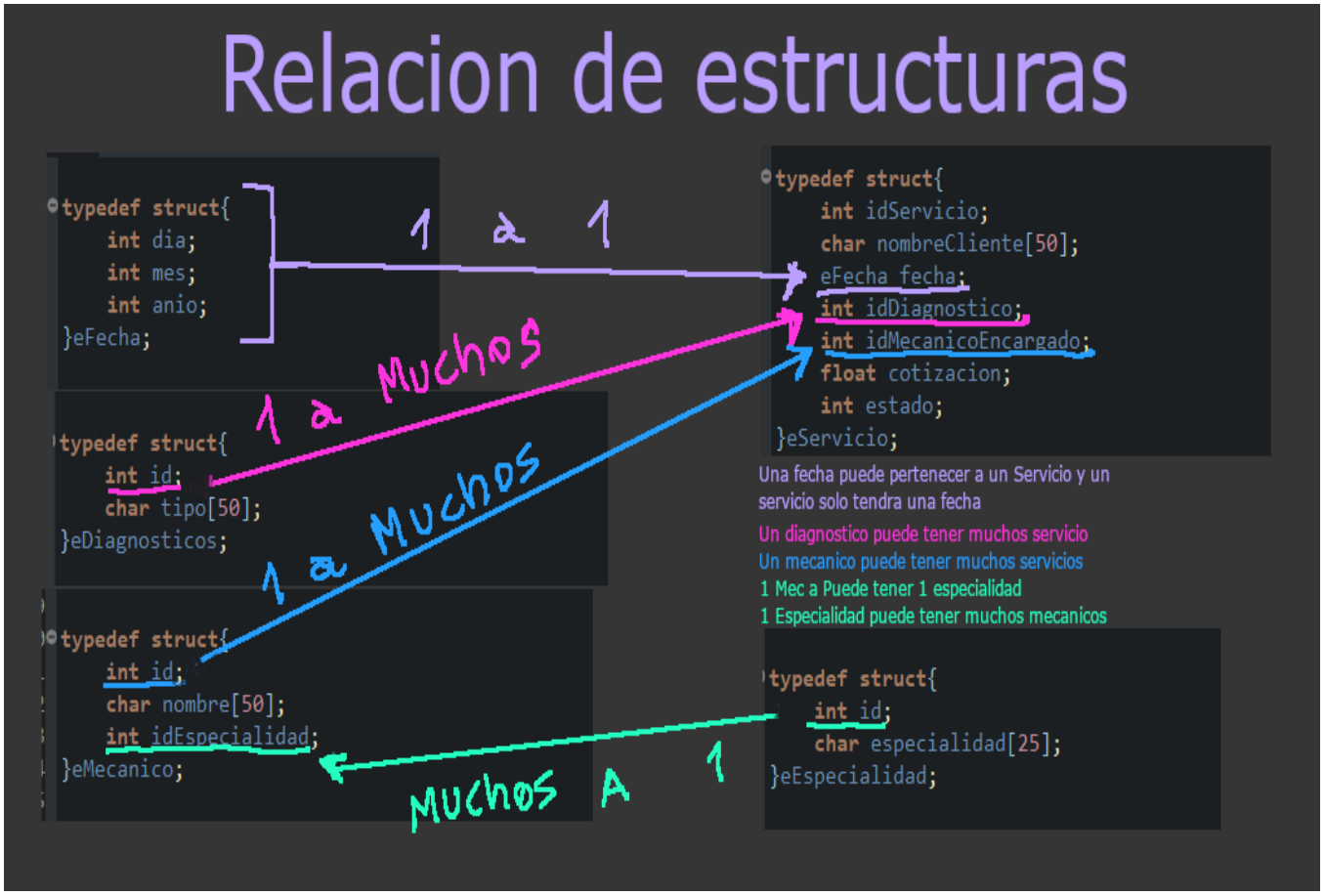
# Recuperatorio 1 Labo 1

---

Apellido: Granadillo  Division: 1B

Nombre: Jeronimo  Legajo: 112415

---




---

Link de la imagen:
https://drive.google.com/file/d/1_Utbf53FT10Rpwg0FiL4F1X7kBpuMbfp/view?usp=sharing

---

Link Del video en drive (Descargar el video) :
https://drive.google.com/file/d/1kSRoAUc2fpOj61jsRwIMTBRqu8xcoo_X/view?usp=sharing

---

/// @brief Esta funcion Busca la existencia de un espacio libre dentro del array listaServicios que es recibido por el parametro llamado del mismo nombre, recorre los indices del array preguntado si el estado es
/// igual a -1
/// @param listaServicios
/// @param sizeS
/// @return retorna el primer indice del array anteriormente mencionado donde se encontro que el estado era -1, sino devuelve -1 que (indica que no hay espacio libre)
**int BuscarEspacioLibre(eServicio listaServicios[], int sizeS);**

/// @brief Esta funcion que se llama desde el main se encarga de, junto a otras funciones cargar un nuevo servicio
/// @param listaServicios
/// @param sizeS
/// @return devuelve si existe la posibilidad de realizar dicha operacion, es decir devuelve -1 si no es posible, y si lo es devuelve un indice de memoria de listaServicios (Esto lo hace llanando a BuscarEspacioLibre)
**int AltaServicio(eServicio listaServicios[],int sizeS);**

/// @brief Verifica la existencia de un servicio existente y Diagnosticado con la funcion BuscarEspacioOcupadoMasServicioDiagnosticado, en base a la respuesta
/// realiza las operaciones para modificar los servicios, despliega la lista de servicios que cumplen con la caracteristica de estar diagnosticados, pide el id de uno de ellos
/// valida su coincidencia y luego despliega un menu donde da las opciones de cambiar nombre del cliente, fecha del servicio, mecanico asignado y salir al menu principal.
/// en base a lo seleccionado se realiza la operacion correspondiente
/// @param listaServicios
/// @param sizeS
/// @param Mecanico
/// @param sizeM
/// @param Diagnostico
/// @param sizeD
/// @param especialidades
/// @param sizeE
/// @return retorna -1 si no es posible realizar la operacion y si se puede retorna un numero correspondiente al indice del array listaServicios devuelto por BuscarEspacioOcupadoMasServicioDiagnosticado
**int ModificarServicio(eServicio listaServicios[],int sizeS,eMecanico Mecanico[],int sizeM,eDiagnosticos Diagnostico[],int sizeD,eEspecialidad especialidades[], int sizeE);**

/// @brief Verifica si existe la posibilidad de dar de baja un servicio llamando a BuscarServicioOpcupado, si es distinto a -1 lo que devuelve la anterior funcion realiza la operacion,
/// esta operacion se realiza pidiendo primero en base a una lista de servicios el id de uno de ellos, una vez obtenido da de baja el servicio de forma logica solo cambiandole el estado de 0 a -1
/// @param listaServicios
/// @param sizeS
/// @param Mecanico
/// @param sizeM
/// @param Diagnostico
/// @param sizeD
/// @return devuelve -1 si no se puede realizar la operacion, en caso de poder devuelve un idice del array listaServicios devuelto por BuscarEspacioOcupado
**int BajaDeServicio(eServicio listaServicios[],int sizeS,eMecanico Mecanico[],int sizeM,eDiagnosticos Diagnostico[],int sizeD,eEspecialidad especialidades[], int sizeE);**

/// @brief Verifica la existencia de servicios y servicios sin diagnosticar llamando a BuscarEspacioOcupadoSinDiagnosticar, si existen servicios, pide el id de un mecanico luego filtra la lista de servicios sin diagnosticar,
/// y pide el id de uno de los servicios, una vez teniendo el id del mecanico y el servicio despliega una lista de diagnosticos, pide seleccionar el id de uno de los diagnosticos y lo carga en la listaServicios.idDiagnostico, tambien
/// pide la cotizacion del mismo y carga la cotizacion y el id del mecanico en la listaServicios.cotizacion y listaServicios.idMecanicoEncargado del servicio seleccionado anteriormente por el id
/// @param listaServicios
/// @param sizeS
/// @param Mecanico
/// @param sizeM
/// @param Diagnostico
/// @param sizeD
/// @param especialidades
/// @param sizeE
/// @return retorna -1 si no se puede realizar la operacion, en caso de poder devuelve un idice del array listaServicios devuelto por BuscarEspacioOcupadoSinDiagnosticar
**int Diagnosticar(eServicio listaServicios[],int sizeS,eMecanico Mecanico[],int sizeM,eDiagnosticos Diagnostico[],int sizeD,eEspecialidad especialidades[], int sizeE);**

/// @brief  Al igual que BuscarEspacioOcupado busca un espacio ocupado dentro del array listaServicios solo que en este caso agrega como condicion que el servicio
/// no este diagnosticado es decir listaServicios.idDiagnostico sea menor a 1
/// @param listaServicios
/// @param sizeS
/// @return devuelve -1 si no se encontro ningun servicio que cumpla la condicion, sino devuelve el indice del primer servicio que cumple dicha condicion.
**int BuscarEspacioOcupadoSinDiagnosticar (eServicio listaServicios[], int sizeS);**

/// @brief hace el calculo de cual es la especialidad mas estudiada segun las especialidades de los mecanicos existentes, para luego mostrar un top 3
/// @param Mecanico
/// @param sizeM
/// @param especialidades
/// @param sizeE
**void EspecialidadesMasEstudidadas(eMecanico Mecanico[],int sizeM,eEspecialidad especialidades[], int sizeE);**

/// @brief Calcula la cantidad total de servicios diagnosticados existentes y asi sabe el 100% de estos, en base a esto, calcula la cantida de servicios diagnosticados que posee cada mecanico
/// con este criterio calcula el porcentaje correspondiente de diagnosticos de cada mecanico en funcion del total de diagnosticos, y los muestra
/// @param listaServicios
/// @param sizeS
/// @param Mecanico
/// @param sizeM
/// @param especialidades
/// @param sizeE
**void PorcentajeDiagnosticosDeCadaMecanico(eServicio listaServicios[],int sizeS,eMecanico Mecanico[],int sizeM,eEspecialidad especialidades[],int sizeE);**