Índice

1. Estructura de Datos

1.1. Incidencias

int	Id_viaje
int	Id_us_registra
int	Id_us_incidencia
char*	Desc_indicencia
int	Est_incidencia

1.2. Pasos

int	Id_viaje
char*	Poblacion

1.3. Usuarios

int	Id_usuario
char*	Nomb_usuario
char*	Localidad
int	Perfil_usuario
char*	User
char*	Login
int	Estado

1.4. Vehiculos

char*	Id_mat
int	Id_usuario
int	Num_plazas
char*	Desc_veh

1.5. Viajes

int	Id_viaje
char*	Id_mat
char*	F_inic
char*	H_inic
char*	H_fin
int	Plazas_libre
int	Viaje
float	Importe
int	Estado

1.6. vIncidencias

Incidencias*	inci
int	tam

1.7. vUsuarios

Usuarios*	user
int	tam

1.8. vVehiculos

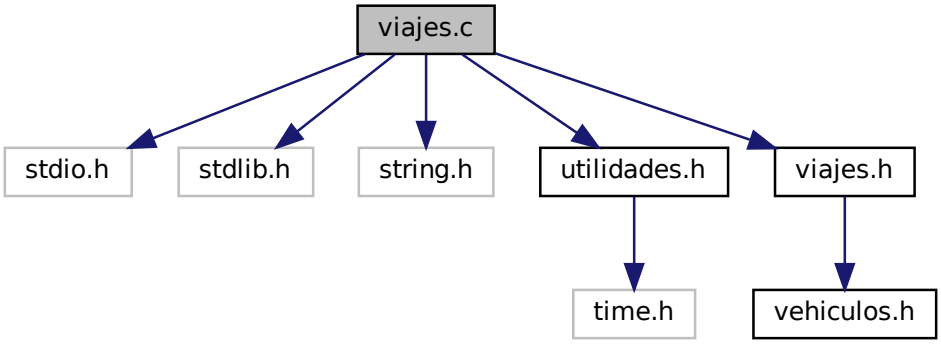
Vehiculos*	vehi
int	tam

1.9. vViajes

Pasos*	pasos
Viajes*	viajes
int	tam_p
int	tam_v

2. VIAJES

2.1. Gráfico de dependencias



2.2. Estructura de Datos

- `struct Viajes`
- `struct Pasos`
- `struct vViajes`

2.3. Funciones

- `Viajes* initViajes(int* n)`
- `Pasos* initPasos(int* n)`
- `void publicarViajeUsuario(vViajes* v,vVehiculos* ve,int userId)`
- `void editarViajesUsuario(vViajes* v,vVehiculos* ve,int userId)`
- `void incorporarseViaje(vViajes* v)`
- `void detalleViaje(vViajes* v)`
- `void publicarViajeAdmin(vViajes* v, vVehiculos* ve)`
- `void eliminarViajesAdmin(vViajes* v)`
- `void modificarViajesAdmin(vViajes* v,vVehiculos *vve)`
- `void listarViajesAdmin(vViajes* v)`
- `void saveViajes(int n,Viajes* viajes)`
- `void savePasos(int n,Pasos* pasos)`
- `int buscarIndexViajes(vViajes* v,int id_viaje)`
- `void listarViajesAbiertos(vViajes* v)`
- `void actualizarViajes(vViajes* v)`

2.4. Definiciones

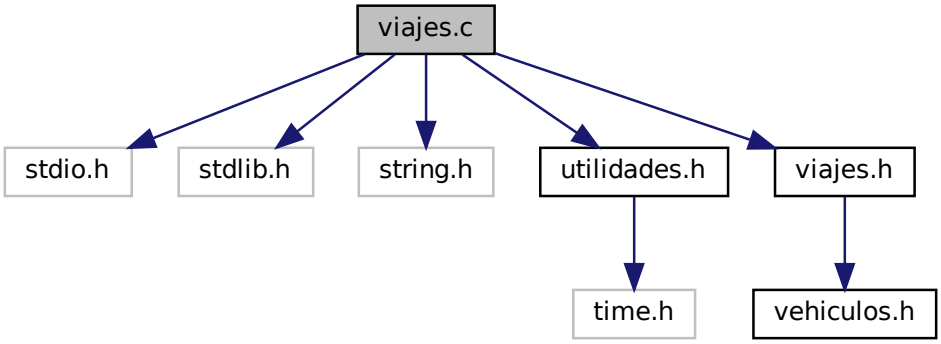
- `Viajes* initViajes(int* n)`
  - **Descripcion**
    - Inicializa una estructura del tipo Viajes.
  - **Parametros**
    - `n` → Referencia al tamaño de la estructura.
  - **Devuelve**
    - Un vector con los datos del fichero Viajes.txt
- `Pasos* initPasos(int* n)`
  - **Descripcion**
    - Inicializa una estructura del tipo Pasos.
  - **Parametros**
    - `n` → Referencia al tamaño de la estructura.
  - **Devuelve**
    - Un vector con los datos del fichero Pasos.txt
- `void publicarViajeUsuario(vViajes* v,vVehiculos* ve,int userId)`
  - **Descripcion**
    - Funcion para publicar un viaje en el sistema de esi-share.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
    - `ve` → Referencia al vector de vehiculos.
    - `userId` → Identificador del usuario que publica el viaje.
- `void editarViajesUsuario(vViajes* v,vVehiculos* ve,int userId)`
  - **Descripcion**
    - Permite la edicion de un viaje en estado abierto.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
    - `ve` → Referencia al vector de vehiculos.
    - `userId` → Identificador del usuario que publico el viaje.
- `void incorporarseViaje(vViajes* v)`
  - **Descripcion**
    - Permite a un usuario incorporarse a un viajes publicado en el sistema.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
- `void detalleViaje(vViajes* v)`
  - **Descripcion**
    - Permite a un usuario ver los datos de un viaje al detalle.
  - **Parametros**
    - `v` → Referencia al vector de viajes.

- `void publicarViajeAdmin(vViajes* v, vVehiculos* ve)`
  - **Descripcion**
    - Permite a un administrador publicar un viaje en nombre de un usuario.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
    - `ve` → Referencia al vector de vehiculos.
- `void eliminarViajesAdmin(vViajes* v)`
  - **Descripcion**
    - Permite a un administrador eliminar un viaje.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
- `void modificarViajesAdmin(vViajes* v,vVehiculos *vve)`
  - **Descripcion**
    - Permite a un administrador eliminar un viaje.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
    - `vve` → Referencia al vector de vehiculos.
- `void listarViajesAdmin(vViajes* v)`
  - **Descripcion**
    - Muestra los viajes al detalle.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
- `void saveViajes(int n,Viajes* viajes)`
  - **Descripcion**
    - Guarda los datos en el fichero Viajes.txt y libera la memoria.
  - **Parametros**
    - `n` → Tamaño del vector user en vViajes.
    - `v` → Referencia al vector de viajes.
- `void savePasos(int n,Pasos* pasos)`
  - **Descripcion**
    - Guarda los datos en el fichero Pasos.txt y libera la memoria.
  - **Parametros**
    - `n` → Tamaño del vector pasos en vViajes.
    - `v` → Referencia al vector de pasos.

- `int buscarIndexViajes(vViajes* v,int id_viaje)`
  - **Descripcion**
    - Busca un viaje el vector vViajes.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
    - `id_viaje` → Identificador del viaje a buscar.
  - **Devuelve**
    - iesima posicion del vector donde se encuentra el viaje.
    - -1 si no se encuentra.
- `void listarViajesAbiertos(vViajes* v)`
  - **Descripcion**
    - Muestra los viajes en estado abierto.
  - **Parametros**
    - `v` → Referencia al vector de viajes.
- `void actualizarViajes(vViajes* v)`
  - **Descripcion**
    - Actualiza el estado de los viajes.
  - **Parametros**
    - `v` → Referencia al vector de viajes.

3. USUARIOS

3.1. Gráfico de dependencias



3.2. Estructura de Datos

- `struct Usuarios`
- `struct vUsuarios`

3.3. Funciones

- `Usuarios* initUsuarios(int * n)`
- `void saveUsuarios(int n, Usuarios *usuarios)`
- `void listarUsuarios(vUsuarios* u,vIncidencias* vi)`
- `void altaUsuario(vUsuarios* v)`
- `void modificarUsuario(vUsuarios* v,int userId)`
- `void perfilUsuario(vUsuarios* v,int userId)`
- `void preguntarIdBaja(vUsuarios* v)`
- `int printPerfil(vUsuarios* v,int userIndex)`
- `void reguntarIdModificar(void)`

3.4. Definiciones

- `Usuarios* initUsuarios(int * n)`
  - **Descripcion**
    - Inicializa una estructura del tipo Usuarios.
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de usuarios.
  - **Devuelve**
    - Un vector con los datos contenidos en el fichero **Usuarios.txt**.

- `void saveUsuarios(int n, Usuarios *usuarios)`
  - **Descripcion**
    - Guarda el contenido acutal de una estructura del tipo Usuarios en ficheros.
  - **Parametros**
    - `n` → Referencia a la posición del vector que almacena el número de usuarios.
    - `usuarios` → Puntero a la estructura de usuarios.
- `void listarUsuarios(vUsuarios* u,vIncidencias* vi)`
  - **Descripcion**
    - Lista el contenido actual de la estructura del tipo Usuarios.
  - **Parametros**
    - `u` → Referencia al vector de Usuarios
    - `vi` → Referencia al vector de Incidencias.
- `void altaUsuario(vUsuarios* v)`
  - **Descripcion**
    - Añade una nueva línea a la estructura del tipo Usuarios.
  - **Parametros**
    - `v` → Referencia al vector de Usuarios.
- `void modificarUsuario(vUsuarios* v,int userId)`
  - **Descripcion**
    - Modificar un línea concreta de la estructura del tipo Usuarios.
  - **Parametros**
    - `v` → Referencia al vector de Usuarios.
    - `userId` → Referencia al entero con el indice del usuario seleccionado.
- `void perfilUsuario(vUsuarios* v,int userId)`
  - **Descripcion**
    - Permite editar al usuario sus datos personales en el sistema y los modifica en la estructura del tipo Usuarios.
  - **Parametros**
    - `v` → Referencia al vector de Usuarios.
    - `userId` → Referencia al entero con el indice del usuario seleccionado.
- `void preguntarIdBaja(vUsuarios* v)`
  - **Descripcion**
    - Pregunta al administrador qué usuario quiere dar de baja.
  - **Parametros**
    - `v` → Referencia al vector de Usuarios.
- `void printPerfil(vUsuarios* v, int userIndex)`
  - **Descripcion**
    - Imprime por pantalla el perfil de un usuario en concreto.
  - **Parametros**
    - `v` → Referencia al vector de Usuarios.
    - `userIndex` → Referencia al entero con el indice del usuario seleccionado.
- `int preguntarIdModificar(void)`
  - **Descripcion**
    - Pregunta al administrador qué usuario desea modificar.
  - **Devuelve**
    - `tmp` → Entero con el id seleccionado.

4. Test

4.1. Test Viajes

4.1.1. Diagrama de flujo