**Cuaderno de profesor**

**Introducción**

En este proyecto hemos realizado un cuaderno de profesor digital en el que hemos dividido los distintos módulos según el tipo de dato que fuese, pensamos que esta sería la mejor opción puesto que aportaría una mayor claridad y facilidad a la hora de la organización del grupo.

El programa te da la opción de gestionar los distintos tipos de datos, entre las cuales se encuentra crearlos, eliminarlos, modificarlos y mostrarlos. Cada tipo de dato hace eso y también vuelca a la estructura desde el fichero o viceversa.

En nuestro grupo cada integrante ha participado en la creación de dos módulos.

**Tecnología**

Hemos usado una serie de herramientas para la realización del programa:

* **Github:** Una plataforma en Internet en la cual subimos el código que vamos haciendo cada semana, también se puede ver el seguimiento de quién ha estado subiendo cada elemento.
* **Git:** Un programa necesario para poder realizar diversas operaciones en github, como subir algo, bajarlo, eliminarlo, etc.
* **Dev-c++:** Una herramienta en la que se ha programado el trabajo, nos sirve para poder escribir, compilar y ejecutar el programa.

**Módulos**

Diagrama

Descripción generada automáticamente

Los distintos colores son simplemente una decoración.

**Cristian Heredia Bravo** se encargó de:

* **Alumnos:** En este módulo realiza la gestión del tipo de dato alumno, pudiendo crear un nuevo alumno, eliminar cualquier alumno, modificar cualquier alumno, mostrar uno o todos los alumnos, encontrar un alumno dependiendo de su id, la cual es única y cargar alumnos desde el fichero a la estructura y viceversa.
* **Horarios:** En este módulo se realiza la gestión del tipo de dato horario, pudiendo crear un nuevo horario, eliminar cualquier horario, modificar cualquier horario, mostrar uno o todos los horarios, encontrar un horario pedido por el usuario y cargar horarios desde el fichero a la estructura y viceversa.

**Descripción de módulos**

**Alumnos**

Este módulo no tiene dependencia de otros módulos, debido a que es uno de los datos más básicos del programa.

El módulo posee una única estructura en la que se va a trabajar:

typedef struct {

char id\_alumno[7];

char nombre\_alum[21];

char direc\_alum[31];

char local\_alum[31];

char curso[31];

char grupo[11];

} alumno;

Esta estructura posee una serie de elementos propios de cada alumno, el id de cada alumno es único.

Ahora pasamos a las diversas funciones que posee el sistema:

---------------------------------------------------------------------------------------------------

//Cabecera: int encontrar\_alumno(alumno \*a, unsigned n, char \*)

//Precondición: vector de alumnos inicializado, la id tiene que ser de 7 carácteres

//Postcondición: Devuelve el número de elemento del alumno, devolverá

-1 si no existe

int encontrar\_alumno(alumno \*a, unsigned n, char \*cid);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

// Cabecera: void mostrar\_alumno (alumno)

// Precondición: alumno inicializado

// Postcondición: Muestra el contenido de alumno

void mostrar\_alumno (const alumno \*a);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: alumno \*leer\_alumnos(unsigned\*);

//Precondición: Ninguna

//Postcondición: Devuelve a través del puntero pasado el número de alumnos e inicializa el vector de alumnos

alumno \*leer\_alumnos(unsigned \*nAlumnos);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void guardar\_alumnos(const alumno \*alumnos, unsigned n)

//Precondición: Ninguna

//Postcondición: Escribe en el fichero Alumnos.txt los n alumnos pasados

void guardar\_alumnos(const alumno \*alumnos, unsigned n);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void menu\_admin\_alumno (alumno \*a, unsigned n)

//Precondición:Ninguna

//Postcondición: Muestra el menu de administrador para alumnos

void menu\_admin\_alumno();

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: int salir\_menu();

//Precondición: Ninguna

//Postcondición: Devuelve 1 si se quiere salir del menú y 0 si no

int salir\_menu();

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void agregar\_alumno(alumno \*\*a, unsigned \*n)

//Precondición: n es el número de alumnos y el vector a tiene que estar inicializado

//Postcondición: Agrega un nuevo alumno introducido por el usuario

void agregar\_alumno(alumno \*\*a, unsigned \*n);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void modificar\_alumno(alumno a)

//Precondición: Ninguna

//Postcondición: Modifica el alumno introducido según quiera el usuario

void modificar\_alumno(alumno \*a);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void mostrar\_alumnos (const alumno \*a, unsigned n)

//Precondición: n es el número de alumnos

//Postcondición: Muestra todos los alumnos del vector

void mostrar\_alumnos (const alumno \*a, unsigned n);

---------------------------------------------------------------------------------------------------

//Cabecera: void quitar\_salto(char \*s)

//Precondición: Ninguna

//Postcondición: Quita de la cadena el carácter '\n'

void quitar\_salto(char \*s);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void eliminar\_alumno(alumno \*\*a, unsigned n, int pos)

//Precondición: n es el número de alumnos y pos es la posición del alumno en el vector

//Postcondición: Elimina el alumno de la posición indicada

void eliminar\_alumno(alumno \*\*alumnos, unsigned \*n,char \*id);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: int existe alumno(alumno \*a, char \*s, unsigned n)

//Precondición: s es la id del alumno a encontrar y n es el número de alumnos

//Postcondición: Devuelve 1 si el alumno existe y 0 si no

int existe\_alumno (alumno \*a, char \*s, unsigned n);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void guardar\_alumno (const alumno \*a, FILE \*f)

//Precondición: Fichero abierto para escritura

//Postcondición: Guarda el alumno en el fichero

void guardar\_alumno(const alumno \*a, FILE \*f);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: int existe\_grupo(alumno \*a, unsigned n, char \*grupo)

//Precondición: Vector de alumnos a inicializado, n es el número de alumnos

//Poscondición: Devuelve 0 si no existe el grupo y 1 si existe

int existe\_grupo(alumno \*a, unsigned n, char \*grupo);

---------------------------------------------------------------------------------------------------

**Módulo horarios**

Horarios tiene dependencia con otros módulos así que usa algunas de sus funciones.

Este módulo usa estructuras de otros módulos, estas son la estructura de alumnos, materias y usuarios. Aunque tiene una estructura propia llamada horario:

typedef struct {

char id\_profesor[4];

int dia\_clase;

int hora\_clase;

char id\_materia[5];

char grupo[11];

} horario;

También posee dos funciones de ámbito privado:

---------------------------------------------------------------------------------------------------

//Cabecera: int grupo\_libre(const horario \*nuevo, const horario \*horarios, unsigned n)

//Precondición: n es el número de horarios, nuevo es el horario recibido que contiene el grupo hora y día que se quiere verificar si está libre

//Poscondición: Devuelve 1 si el grupo está libre a esa hora y día o 0 si no está libre

int grupo\_libre(const horario \*nuevo, const horario \*horarios, unsigned n);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: int profesor\_libre(const horario \*nuevo, const horario \*horarios, unsigned n)

//Precondición: n es el número de horarios, nuevo es el horario recibido que contiene el profesor (id), hora y día que se quiere verificar si está libre

//Poscondición: Devuelve 1 si el profesor está libre en esa fecha o 0 si no.

int profesor\_libre(const horario \*nuevo, const horario \*horarios, unsigned n);

---------------------------------------------------------------------------------------------------

Ahora pasamos a las funciones públicas:

---------------------------------------------------------------------------------------------------

//Cabecera: void menú\_admin\_horarios()

//Precondición: Ninguna

//Poscondición: Muestra el menú de administrador para horarios

void menu\_admin\_horarios();

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void agregar\_horario(horario \*\*hor, usuario \*usuarios, materia \*materias, alumno \*alumnos, unsigned nUsuarios, unsigned nMaterias, unsigned nAlumnos, unsigned \*nHorarios)

//Precondición: Todos los vectores deben estar inicializados, cada unsigned son el número de los distintos elementos de los vectores

//Poscondición: Agrega un horario teniendo en cuenta que exista los datos compartidos en otros ficheros

void agregar\_horario(horario \*\*hor, usuario \*usuarios, materia \*materias, alumno \*alumnos, unsigned nUsuarios, unsigned nMaterias, unsigned nAlumnos, unsigned \*nHorarios);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void eliminar\_horario(horario \*\*hor, int encontrado, unsigned \*nHorarios)

//Precondición: encontrado es la posición del horario que se desea eliminar y nHorarios es el número de horarios que hay

//Poscondición: Elimina un horario solicitado por el usuario.

void eliminar\_horario(horario \*\*hor, int encontrado, unsigned \*nHorarios);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void modificar\_horario(horario \*hor, materia \*materias, usuario \*usuarios, alumno \*alumnos, unsigned \*nHorarios, unsigned nMaterias, unsigned nUsuarios, unsigned nAlumnos, int encontrado);

//Precondición: Todos los vectores deben estar inicializados, cada unsigned son el número de los distintos elementos de los vectores, encontrado es el horario que se desea modificar.

//Poscondición: Modifica un horario solicitado por el usuario.

void modificar\_horario(horario \*hor, materia \*materias, usuario \*usuarios, alumno \*alumnos, unsigned \*nHorarios, unsigned nMaterias, unsigned nUsuarios, unsigned nAlumnos, int encontrado);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void mostrar\_horarios(horario \*horarios, unsigned nHorarios)

//Precondición: nHorarios es el número de horarios en el vector

//Poscondición: Muestra todos los horarios del vector

void mostrar\_horarios(horario \*horarios, unsigned nHorarios);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void guardar\_horarios(const horario \*horarios, unsigned n)

//Precondición: n es el numero de horarios y horarios es el vector en sí

//Poscondición: Guarda en el fichero todos los horarios

void guardar\_horarios(const horario \*horarios, unsigned n);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void guardar\_horario (const horario \*h, FILE \*f)

//Precondición: Fichero abierto para escritura

//Poscondición: Guarda en el fichero el horario recibido

void guardar\_horario (const horario \*h, FILE \*f);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: horario \*leer\_horarios(unsigned \*nHorarios)

//Precondición: Ninguna

//Poscondición: Inicializa el vector de horarios y almacena en nHorarios el número de horarios

horario \*leer\_horarios(unsigned \*nHorarios);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void mostrar\_horario(const horario \*hor)

//Precondición: Ninguna

//Poscondición: Muestra por pantalla el horario recibido

void mostrar\_horario(const horario \*hor);

---------------------------------------------------------------------------------------------------

---------------------------------------------------------------------------------------------------

//Cabecera: void mostrar\_horarios(horario \*horarios, unsigned nHorarios)

//Precondición: nHorarios es el número de horarios que existen

//Poscondición: Muestra por pantalla el vector de horarios

void mostrar\_horarios(horario \*horarios, unsigned nHorarios)

---------------------------------------------------------------------------------------------------

**Plan de Pruebas**

En este apartado veremos algunas funciones de algunos de los módulos.

Empezamos con la función de agregar\_alumnos:

Texto

Descripción generada automáticamente

En este caso recibimos un puntero a un vector de alumnos y un puntero de unsigned (número de alumnos). El condicional se cumplirá si la función existe\_alumno devuelve un 0. Si se cumple el condicional se crea un nuevo alumno en la posición n del vector. La función de existe\_alumno es la siguiente:

Texto

Descripción generada automáticamente

En cada interacción del bucle se comprueba si la id pasada es la misma que la de cada id de cada alumno, en el caso que sea el mismo se devolverá un 1.

Ahora la función de eliminar\_horario:

Texto

Descripción generada automáticamente

En esta función se realiza un bucle que se repite desde encontrado hasta nHorarios-1, encontrado puede tener valores desde 0 hasta nHorarios-1, no se puede obtener otro valor porque en el vector no hay valores negativos ni superiores a nHorarios-1. El máximo valor que tendrá i será nHorarios -2, e i+1 tendrá el valor de nHorarios-1, el tamaño del vector es nHorarios y la última posición del vector es nHorarios-1, entonces en la última interacción será igualar el valor de la posición de nHorarios-2 a nHorarios -1.