UNIVERSIDAD DISTRITAL FRANCISCO JOSÉ DE CALDAS

FACULTAD DE INGENIERÍA

PROYECTO CURRICULAR DE INGENIERÍA DE SISTEMAS

Integrantes:

Diego Alejandro Bautista Castañeda - 20192020139
Steven Espejo Cabarcas - 20192020138

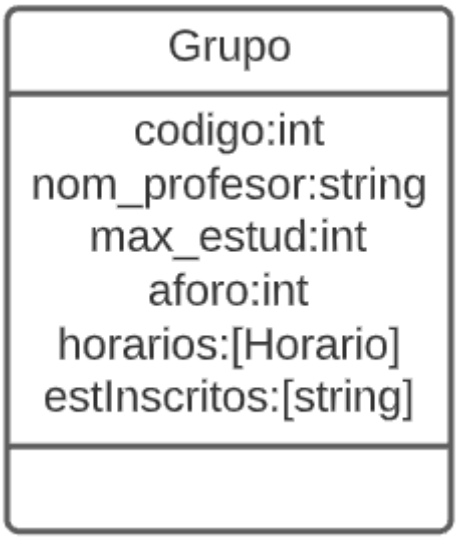
DOCUMENTACIÓN DE SOPORTE
PROYECTO FINAL

BOGOTÁ D.C
2022

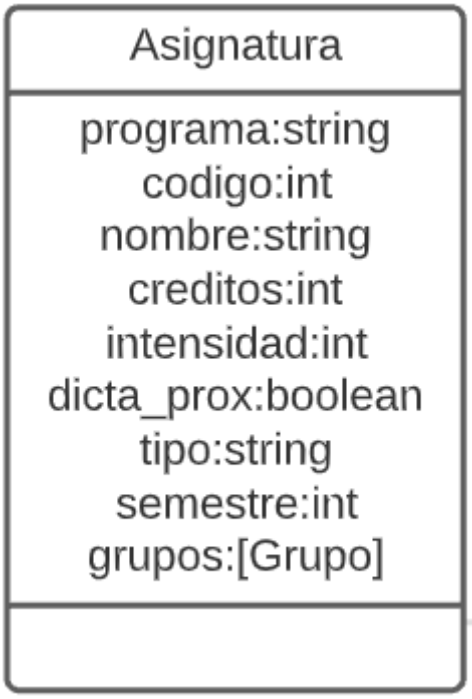
ESTRUCTURAS:

Se definieron principalmente las siguientes estructuras, definidas en “estructura.h”:

Grupo: Información de los grupos de las asignaturas

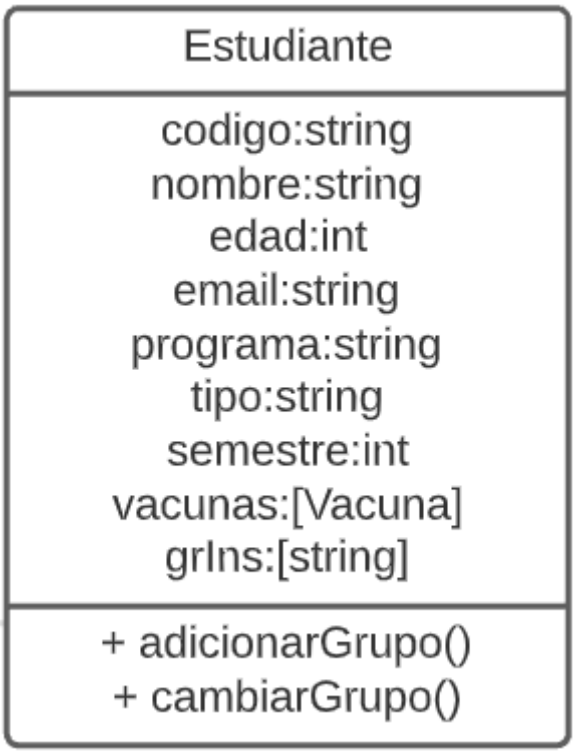


Asignatura: Información de las asignaturas

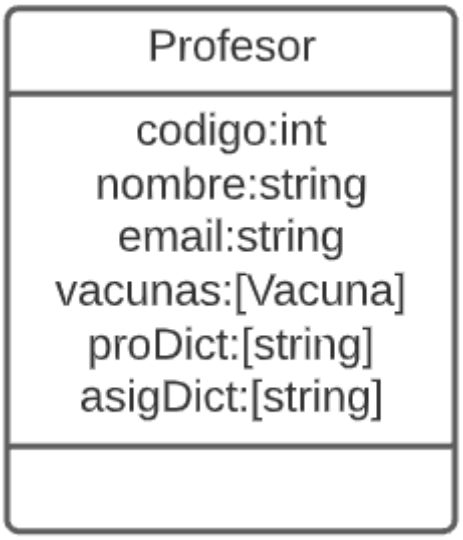


Horario:Información de los horarios de una asignatura.

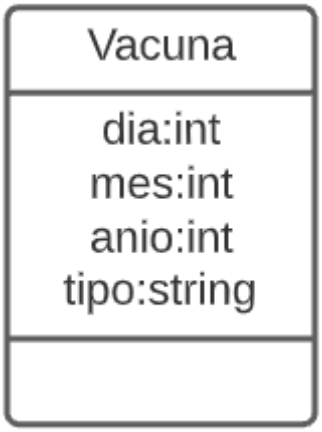
Estudiante: Información de cada uno de los estudiantes, con sus respectivas funciones de adición y modificación de grupos inscritos.



Profesor:Información de los profesores de la facultad de Ingeniería.



Vacuna: Información de las vacunas recibidas en un Profesor o Estudiante.



CLASES:

- **Clase Programa,** definida de la siguiente manera**:**

```
template <class T>
class Programa{
  nodo<T> *sistemas, *electronica, *industrial, *electrica, *catastral;
  int tam_sistemas, tam_electronica, tam_industrial, tam_electrica, tam_catastral;

  public: Programa(){
    tam_sistemas=0, tam_electronica=0, tam_industrial=0, tam_electrica=0, tam_catastral=0;
    sistemas=NULL, electronica=NULL, industrial=NULL, electrica=NULL, catastral=NULL;
  }

  string upper(string s);
  void insertar(T nueva_info);
  T obtener(string programa, int posicion);
  int tam(string programa);
  void validarInsercion(nodo<T> *&cab, nodo<T> *nuevo);
  T validarObtener(nodo<T> *&cab, int pos);
};
```

Declarando cada una de las cabeceras para los respectivos programas de la facultad, además, de sus debidos tamaños.

Utilizando para la clase la siguiente función:

string upper(string s)**:** Utilizada para hacer validación del programa insertado.

- **Clase Semestre,** definida de la siguiente manera**:**

```
template <class T>
class Semestre{
  nodo<T> *uno, *dos, *tres, *cuatro, *cinco, *seis, *siete, *ocho, *nueve, *diez;
  int tam_uno, tam_dos, tam_tres, tam_cuatro, tam_cinco, tam_seis, tam_siete,
tam_ocho, tam_nueve, tam_diez;

  public: Semestre(){
    tam_uno=0, tam_dos=0, tam_tres=0, tam_cuatro=0, tam_cinco= 0, tam_seis=0,
tam_siete=0, tam_ocho=0, tam_nueve=0, tam_diez=0;
    uno=NULL, dos=NULL, tres=NULL, cuatro=NULL;
  }

  void insertar(T nueva_info);
  T obtener(int semestre, int posicion);
  int tam(int semestre);
  void validarInsercion(nodo<T> *&cab, nodo<T> *nuevo);
  T validarObtener(nodo<T> *&cab, int pos);
};
```

Declarando cada una de las cabeceras para los respectivos semestres, además, de sus debidos tamaños.

- **Clase Intensidad,** definida de la siguiente manera**:**

```
template <class T>
class Intensidad{
  nodo<T> *dos, *tres, *cuatro, *cinco, *seis;
  int tam_dos, tam_tres, tam_cuatro, tam_cinco, tam_seis;

  public: Intensidad(){
    tam_dos=0, tam_tres=0, tam_cuatro=0, tam_cinco=0, tam_seis=0;
    dos=NULL, tres=NULL, cuatro=NULL, cinco=NULL, seis=NULL;
  }

  void insertar(T nueva_info);
  T obtener(int intensidad, int posicion);
  int tam(int intensidad);
  void validarInsercion(nodo<T> *&cab, nodo<T> *nuevo);
  T validarObtener(nodo<T> *&cab, int pos);
};
```

Declarando cada una de las cabeceras para cada una de las intensidades pertenecientes a una asignatura, además, de sus debidos tamaños.

- **Clase Créditos,** definida de la siguiente manera**:**

```
template <class T>
class Creditos{
  nodo<T> *uno, *dos, *tres, *cuatro;
  int tam_uno, tam_dos, tam_tres, tam_cuatro;

  public: Creditos(){
    tam_uno=0, tam_dos=0, tam_tres=0, tam_cuatro=0;
    uno=NULL, dos=NULL, tres=NULL, cuatro=NULL;
  }

  void insertar(T nueva_info);
  T obtener(int creditos, int posicion);
  int tam(int creditos);
  void validarInsercion(nodo<T> *&cab, nodo<T> *nuevo);
  T validarObtener(nodo<T> *&cab, int pos);
};
```

Utilizando las siguientes funciones para las clases anteriormente descritas:

1. ***void insertar(T nueva_info)*****:** Insertar un elemento en la lista
2. ***T obtener(string programa, int posicion)*****:** Obtener un elemento según el programa y según la posición.
3. ***int tam(string programa)*****:** Obtener el tamaño de la lista.
4. **void validarInsercion(nodo<T> *&cab, nodo<T> *nuevo):** Validación en el momento de ingresar un nuevo elemento dentro de la lista.
5. **T validarObtener(nodo<T> *&cab, int pos):** Validación en el momento de querer obtener un elemento dentro de la lista.

Librerías empleadas:

Lista.h:

```
template <class T>
class Lista{
  nodo<T> *cab;
  int tam;

  public: Lista(){
    cab = NULL;
    tam=0;
  }

  void insertar(T nueva_info);
  T obtener(int posicion);
  int tamLista();
  void editar(T nueva_info, int posicion);
};
```

Empleada principalmente para poder guardar la información de cada una de las estructuras:

- Profesores
- Asignaturas
- Estudiantes
- Grupos

Utilizando las siguientes funciones**:**

1. ***void insertar(T nueva_info)*****:** Insertar un elemento en la lista.
2. ***T obtener(int posicion)*****:** Obtener un elemento según la posición.
3. ***int tamLista()*****:** Obtener el tamaño de la lista.
4. **void editar(T nueva_info, int posicion):** Editar un registro pasando la nueva información y según la posición en donde se encuentra el que se va a editar.

Interna.h:

```
template <class T>
class Interna{
  nodoL<T> *cab;
  int tam;

  public: Interna(){
    cab = NULL;
    tam=0;
  }

  void insertar(T nueva_info);
  T obtener(int posicion);
  int tamLista();
  int obtenerPosicion(string cod);
  void editar(T nueva_info, int posicion);
  bool encontrar(string cod);
};
```

Empleada para poder guardar la información de estructuras internas pertenecientes para una estructura de Lista.h:

- Vacunas
- Horarios

Utilizando las siguientes funciones**:**

1. ***void insertar(T nueva_info)*****:** Insertar un elemento en la lista.
2. ***T obtener(int posicion)*****:** Obtener un elemento según la posición.
3. ***int tamLista()*****:** Obtener el tamaño de la lista.
4. **int obtenerPosicion(string cod):** Obtener la posición según un código .
5. **void editar(T nueva_info, int posicion):** Editar un registro pasando la nueva información y según la posición en donde se encuentra el que se va a editar.
6. **bool encontrar(string cod):** Saber si el código se encuentra dentro de la lista o no.

Memoria Secundaria:

La información guardada se obtiene a partir de archivos planos ubicados en la carpeta “archivos”.
Los cuales son:

- asignatura.txt
- estudiantes.txt
- profesores.txt
- grupos.txt

Main:

Librerías implementadas:

```
#include <iostream>
#include <stdlib.h>
#include <string>
#include <fstream>
#include <typeinfo>
#include "estructura.h"
#include "Lista.h"
#include "Intensidad.h"
#include "Programa.h"
#include "Creditos.h"
#include "Semestre.h"
using namespace std;
```

Variables:

```
//Listas generales
Lista<Asignatura> asignaturas;
Lista<Profesor> profesores;
Lista<Estudiante> estudiantes;
Lista<Grupo> grupos;

//Listas de cabeceras
Programa<Asignatura> progAsig;
Intensidad<Asignatura> intAsig;
Creditos<Asignatura> creAsig;
Semestre<Asignatura> semAsig;
```

Funciones usadas:

- **void regisGrupo(Asignatura asig_par):** Registrar grupo individualmente.

- **Grupo registrarGrupo(Asignatura asig_par):** Función para registrar grupo, definiéndolo en el momento del registro de una Asignatura.

- **bool registrarAsignatura():** Función para registrar una asignatura.

- **bool registrarProfesor():** Función para registrar un profesor.

- **bool registrarEstudiante():** Función para registrar un estudiante.

- **void escribirAsig():** Función para escribir una asignatura en el archivo plano “asignatura.txt”.

- **void escribirEstud():** Función para escribir un estudiante en el archivo plano “estudiantes.txt”.

- **void escribirProf():** Escribir un profesor en el archivo plano “profesores.txt”.

- **void escribirGrupo():** Función para escribir un grupo en el archivo plano “grupos.txt”.

Se realizó una escritura previa en los archivos planos, y realizar la debida lectura de cada uno de ellos:

- **void lecturaProf():** Lectura de los profesores.
- **void lecturaEstud():** Lectura de los estudiantes.
- **void lecturaGrupo():** Lectura de los grupos.
- **void lecturaAsig():** Lectura de las asignaturas.

Funciones para cada una de las consultas:

- **void consulta1():** Se validó el número total de asignaturas por cada programa, evidenciando el tipo de programa (Pregrado y Postgrado), el semestre y número de créditos de cada asignatura.

- **void consulta2():** Nos ofrece una lista de asignaturas que se dictarán el próximo semestre, donde se indica el código, nombre, horario y cupos disponibles.

- **void consulta3():** Nos ofrece un listado de estudiantes que están inscritos en un grupo y cuántas vacunas tiene.

- **void consulta4():** Se valida que asignaturas tiene inscritas un estudiante, además, de evidenciar las vacunas que tiene aplicada con su respectiva fecha.

- **void consulta5():** Valida el número de estudiantes vacunados con una dosis específica de cada facultad, evidenciando el tipo de programa, semestre y fecha en que fue aplicada la última vacuna.

- **void consulta6():** Nos ofrece un listado de profesores que dictan en un programa en específico, mostrando las asignaturas que este dicta, el tipo de vacuna y la fecha de cada una de sus vacunas.

- **int esBis(int anio):** Esta función valida si un año es bisiesto o no.

- **int dMes(int mes, int anio):** Esta función nos dice cuántos días tiene cada mes dependiendo si el año dado es bisiesto.

- **int calcularDias(int dia1,int mes1,int anio1, int dia2,int mes2,int anio2):** Esta función se encarga de calcular cuántos días hay de una fecha a otra con ayuda de las dos anteriores y de esta manera permitirnos realizar la consulta número 7.

- **void consulta7():** Se listan los profesores y estudiantes en donde su segunda fecha de vacunación tenga más de 6 meses de

diferencia con la fecha actual y no se haya aplicado la tercera, además, de indicar cargo y programa al que pertenece cada uno.

- **void consulta8():** Se listan los salones que están ocupados en una hora y día exactos, mostrando la asignatura, el programa, el nombre del profesor que dicta, la capacidad máxima del salón y el número de estudiantes inscritos en la asignatura.

- **void consulta9():** Dada una vacuna, mostrar el listado de estudiantes y profesores que han sido inoculados con ella, indicando la fecha y en cuál o cuáles dosis fue recibida.

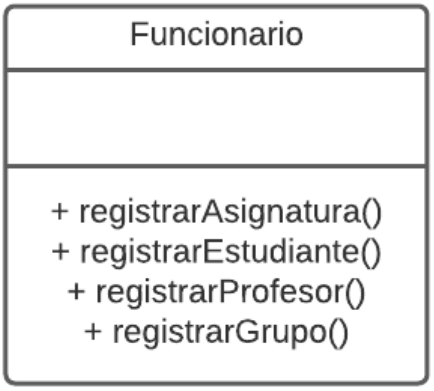
- **void InsEstud():** Adicionar un grupo al estudiante, según su programa y su semestre.

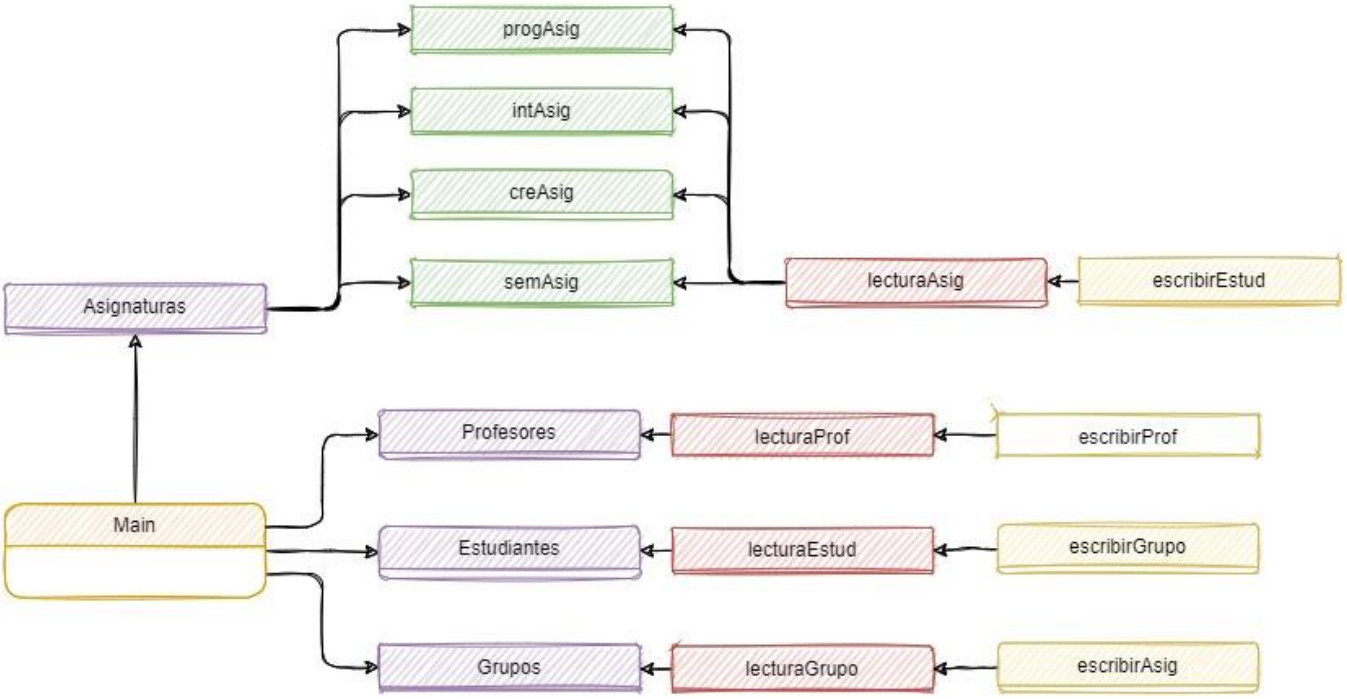
- **void modEstud():** Poder modificar los grupos de las asignaturas que el estudiante tiene inscritas.

- **void menuMaterias():** Menú en el cual se muestran las opciones de inscribir o modificar un grupo.

- **void menuConsultas():** Menú para saber que consulta desea visualizar el usuario.

- **void menuRegistrar():** Menú para realizar cada una de las funciones mostradas de registro.




progAsig
intAsig
creAsig
semAsig
Asignaturas
lecturaAsig
escribirEstud
Profesores
lecturaProf
escribirProf
Main
Estudiantes
lecturaEstud
escribirGrupo
Grupos
lecturaGrupo
escribirAsig