PONTIFICIA UNIVERSIDAD
JAVERIANA DEPARTAMENTO DE
INGENIERÍA

ESTRUCTURAS DE DATOS

PROYECTO
TERCERA
ENTREGA

INTEGRANTES:
SANTIAGO CHAUSTRE
JAVIER MARÍN

PROFESOR:
GABRIEL
AVILA

04 DE MAYO DEL 2017
BOGOTÁ
COLOMBIA

En el siguiente documento se encuentra el diseño de la primera entrega del proyecto de semestre el cual se trata de recrear el SIU de la universidad, pero para esto solo vamos a tener en cuenta los estudiantes y las clases en estas contaran con pre-requisistos que son asignaturas.

En esta entrega se nos pide corregir la primera entrega y ya con esta hacer tres funciones mas las cuales son:

1- Cargar la información de los estudiantes y con esto poder crear un grafo con esta información
2- Poder imprimir el grado de separación que tienen dos estudiantes.
3- Tener todas las funciones ejecutándose en su perfección de la entrega anterior.

Para ello nosotros decidimos tener los siguientes TADs. En este documento se van ha obviar los geters y seters para poder evitar la redundancia.

TAD SAE

Conjunto mínimo de datos

- Listasemestres<llista de semestres> Es la lista de los semestres que se tienen en el sistema

- Departamentos<lista de departamentos> Es la lista de los Departamentos que se tienen en el sistema

- aulas<llista de aulas> Es la lista de las aulas que se tienen en el sistema

- prerequisitos<multi mapas> Es el multimap de los pre-requisitos de las asignaturas.
- RedSocial<Grafo>Es el grafo de apontudares de Estudiantes y clases.

Comportamiento (Operaciones) del objeto

- Verificar Aula,<VerificarAula()>,booleano, verifica si existe una aula en especifico para esto se le manda como parámetro el id del aula y este retorna si la encontró o no.
- Agregar Asignatura<AgregarAsignatura()> Agrega asignaturas al sistema y este recibe la información necesaria para hacer esto.
- Verificar Semestre<VerificarSemestre()>apuntador de Semestre,verifica si existe un semestre en concreto y si es así retorna su apuntador.
- Realizar Clase por asignatura<RealizarClaseAsign()> genera las clases por asignatura dependido obviamente cada asignatura.
- Verificar departamento<VerificarDept()>apuntador de Departamento, verifica si existe un departamento en concreto por su nombre y si es así retorna su apuntador.
- Realizar Estructura por clase<RealizarEstuclase()>booleano, crea la estructura de cada clase y retorna un booleano para ver si se creo efectivamente o no
- Verificar departamento<VerificarDept2()>apuntador de Departamento, verifica si existe un departamento en concreto por su codigo y si es así retorna su apuntador.
- Horario de estudiante<horarioestud()>oraciones, genera el horario de un estudiante para poderlo imprimir por pantalla.
- Convertir hora<convertirhora()>oraciones, convierte las horas de cada clase a un formato de impresión por pantalla.
- Comparar hora<compararhora()>booleano, verifica cual de las dos horas que se le mandan como parámetro cual es menor.
- Siguiente hora <siguientehora()> oraciones, devuelve la siguiente hora para el momento de la impresión del horario de un estudiante.
- Agregar tupla <AgregarTupla()>booleano, agrega una tupla de dos identificaciones y retorna un booleano para ver si fue agregada exitosamente o no.

- Verificar asignaturas <VerificarAsignaturas()>apuntador de asignatura, verifica si una asignatura en especifico existe o no y si esta llega a existir devuelve el apuntador de la signatura.
- Demanda de asignatura <DemandaAsign()> numero, calcula la demanda de una asignatura en especifico en un semestre en especifico.
- Ultimo Semestre<ultimoSemestre()>oraciones, revisa cual es el ultimo semestre a nivel cronológico fue registrado.
- Demanda estudiante<Demandaestud()>calcual la demanda de un estudiante en el resto de su carrera.
- Siguiente semestre <siguienteSemestre()> oraciones, revisa cual es el siguiente semestre en el sistema.
- Verificar si es Prerrequisito <verificarsiesPrerrequisito()>booleano, verifica si una asignatura en especifico es pre-requisito de otra.
- Es prerrequisito<esprerrequisito()>booleano, verifica si una asignatura en especifico es pre-requisito de aulguna otra.
- Eliminar Innecesarios <eliminarInnecesarios()>, elimina las asignaturas repetidas en el sistema.
- Estan Pre-requisitos <estanPrerreq()> revisa si una asignatura en especifico tiene pre-requisitos o no.
- MakeRedSocial<MakeRedSocial()>genera un grafo de estudiantes dependiendo un semestre que se recibe como parametro.
- printSocial<printSocial()> imprime el grafo que se creo.
- GradoSeparacion<GradoSeparacion()> Recibe el nombre de dos estudiantes y el grado de separacion que necesita y retorna los amigos que tienen en esta separación.

  TAD Archivo

  Conjunto mínimo de datos

- arch <flujo de datos>, Es el nombre del archivo a leer.

- lineact<oraciones>, Es la linea que se le acaba de leer del archivo anteriormente mensionado.
- sae<apuntador de la clase SAE>, Es el apuntador que comunica con el sae del sistema.

Comportamiento (Operaciones) del objeto

- Archivo<Archivo()>,Es el constructor de la clase.

  - Siguiente linea <Nextline()>,oraciones,devuelve la siguiente linea del archivo
  - separar Palabra<separarPalabra()>separa todas la información con un primer carácter de separación.
  - separarPalabra2<SepararPalabra2()>separa la información con un segundo carácter de separación.
  - Llenar Archivo<llenarArch()>,ordena las funciones anteriormente mensionadas.
  - Llenar prerequisito<llenarprereq()>, genera las asignaturas que son prerequisitos
  - llenarArch2<llenarArch2()>,ordena las funciones anteriormente mensionadas.
  - Archivo abierto<archopen()>, booleano, verifica si el flujo de datos con el archivo esta abierto.
  - Archivo cerrado<archclose()> cierra el flujo de datos con el archivo.

TAD Sesion

Conjunto mínimo de datos

- HoraInicioClase<oraciones>Es la hora de inicio de la clase.
- HoraFinClase< oraciones>Es la hora en que la clase termina.
- TipoSesion<oraciones> Es el tipo de sesion que se tiene por ejemplo si dura las 18 semanas o llega durar 22 semanas.

- Dia<oraciones>Es el dia en que se dicta la clase.
- IdAula<oraciones>Es el codigo para identificar el aula en la que se dicta la sesion de clase.

Comportamiento (Operaciones) del objeto

- como se menciono anteriormente se van a obviar los geters y seter para evitar la redundancia y la brevedad de este documento.

TAD Semestre

Conjunto mínimo de datos

- CicloElectivo<oraciones>Es el ciclo electivo del semestre.
    - estudiantes<lista>Es la lista de estudiantes que están matriculados para el

  semestre. Comportamiento (Operaciones) del objeto
- Verificar Estudiante<VerificarEstudainte()>Esta función sirve para poder observar si se tiene un estudiante ya registrado.
- Verificar Estudiante 2<VerificarEstudainte2()>,apuntador de Estudiante,Esta función sirve para poder observar si se tiene un estudiante ya registrado y devuelve el apuntador del estudiante.

TAD Estudiante

Conjunto mínimo de datos

- nombre<oraciones>,es el nombre del estudiante registrado.
- Idestud<oración>, Es el código único del estudiante en el sistema.
- apellido<oración>, Es el apellido del estudiante registrado.
- GradoAcademico<oración> Es el grado académico en el cual el estudiante esta matriculado.
- Grado<oración>Es el grado que tiene el estudiante en el momento del estudio.

- CdProgAcadBase<oración>Es el código del programa en el que el estudiante se encuentra matriculado.
- cd_actual_baja<oración>Es el código de baja que se le dio al estudiante.
- NombreActualPrograma<oración> Es el nombre del programa en el que el estudiante se encuentra matriculado.
- clases<lista>Es la lista de las notas que tiene el estudiante por clase.

Comportamiento (Operaciones) del objeto

- Agregar notas a la clase <AgregarClasenotas()>, se agregan las notas a una clase en especifico.
- Obtener Nombre Completo <GetNombreComp()>,oraciones, retorna el nombre y apellido de un estudiante.
- Verificar notas de clase<verificarclasenota()>booleano, verifica si se tiene notas una clase en especifico.

TAD Departamento

Conjunto mínimo de datos

- Nombredept<oración>, Es el nombre del departamento que se tienen registrados en el sistema.
- Oracd<oración>, Es el codigo con el que se identifica el departamento dentro del sistema.
- Catalogo<oración>,Es el codigo del catalago en la que se encuentra el Departamento.
- Asignaturas<lista de apuntadores de la asignatura>Es la lista de los apuntadores de las asignaturas que tiene el departamento.

Comportamiento (Operaciones) del objeto

- Agregar Asignatura<AgregarAsignatura()>Esta función agrega las asignaturas a correspondientes al departamento.
- Verificar Asignatura<VerificarAsignatura()>Esta función verifica si una asignatura ya se encuentra en el sistema y si es así la retorna.

TAD ClaseXestudiante

Conjunto mínimo de datos

- estadoinscrp<oración> Es el estado de la clase.
- clase<Clase>Es la clase que relaciona la nota y el estudiante.
- Nota<numero>, Es la nota del estudiante en la clase que tiene relación.
- creditosinscrp<numero>, la cantidad de creditos que vale la materia.
- Sistemacalf <oración>, es el sistema de calificación de la calse.
- Finscrp<oración>, muetra si la clase esta inscrita.
- fretiro<oración>, muetra si la calse se retiro.

Comportamiento (Operaciones) del objeto

- como se menciono anteriormente se van a obviar los geters y seter para evitar la redundancia y la brevedad de este documento.

TAD Clase

Conjunto mínimo de datos

- nClase< oración >, Es el numero de la clase con el cual se usa para identificarla
- EstadoClase <oracón>, Es el estado de la clase ya sea “Activo” o “Desactivo”.
- EstadoInscripcion <oracón>, Cual es el estado de la clase en términos de inscripción
- FFinal <oracón>, La fecha de cierre que tiene la clase.
- Fincial <oracón>, La fecha en la cual la clase empieza.
- TotalInscritos <oracón> El numero de los estudiantes inscritos a la clase.
- ListaSesion<lista> La lista de sesiones con la que cuenta la clase.
- Seccion<numero>, Es el código de la sección en la que pertenece la clase.
- Cicloelectivo<oración>Es el ciclo electivo en la que pertenece la clase.
- aulasoli<numero>, Es el numero de la aula solicitada.
- capacidadinscrp<numero>, la cantidad de inscritos en la clase.
- Descrp<oración>, es la descripción de la clase.
- Nombreclas<oración>, es el nombre de la clase.
- idasign<oración>, es la identificación de la asignatura a la que pertenece la clase.

Comportamiento (Operaciones) del objeto

- AgregarSesion<AgregarSesion()>Esta función agrega una nueva sesion a la lista de la clase.

TAD Aula

Conjunto mínimo de datos

- IdAula<oración> Es el código único asignado al aula.
- Capacidad_aula<numero> Es el numero de puestos que se encuentran en el aula.

Comportamiento (Operaciones) del objeto

- como se menciono anteriormente se van a obviar los geters y seter para evitar la redundancia y la brevedad de este documento.

TAD Asignatura

Conjunto mínimo de datos

- listaClase<lista>, Es la lista de clases que tiene asignado la asignatura.
- IdCurso<oración>El codigo unico que se le es asignado a la asignatura en el sistema.
- NombreAsignatura<oración>, El nombre de la asignatura.
- Grado<oración>El grado de la asignatura por ejemplo: “pregrado”,”posgrado”

Comportamiento (Operaciones) del objeto

- AgregarClase<AgregarClase()>, Esta funcion es para poder crear las clases que tiene la asignatura.
- Verificar clase<VerificarClase()>, apuntador de clase,Verifica si la clase esta ingresada en el sistema y si es así retorna el apuntado de la clase.

TAD grafo

Conjunto mínimo de datos.

- Vertexs <multi-mapa> Es donde se van a guardar todos los vertices con sus adyacencias en el sistema.
- type<Boleano>Es el que describe si el grafo es dirigido o no.

Comportamiento (Operaciones) del objeto

- addVertex<addVertex()> recibe un dato y con este crea y agrega el vertice al grafo.

- eraseEdge<eraseEdge()>recibe un camino y lo borra del sistema.
- findVertex<findVertex()> retorna un vertice dependiendo del dato que se le ingresa como parametro.
- addEdge<addEdge()>crea la adyacencia entre dos vertices y le asigna un peso.
- eraseVertex<eraseVertex()> recibe un vértice y lo borra del sistema.
- flatTravel<flatTravel()>imprime todos los caminos que existen.
- printAll<printAll()>imprime todos los vertices con su lista de adyacencia.
- resetVisited<resetVisited()>modifica el atributo de visto en todo los vertices.
- isPath<isPath()>mira si hay camino de euler.
- EulerPath<EulerPath()> imprime el camino de Euler si existe.
- validNext<validNext()>mira si el siguiente vertice esta vistado o no.
- numberofConectedComponents<numberofConectedComponents()> es el numero de componentes conectados a un vertice.
- resetValue<resetValue()>reinicia el valor de los vertices.
- dijkstra<dijkstra()>Genera el camino de dijkstra y lo imprime.
- restEdgeVisted<restEdgeVisted()>modifica el atributo de visto en todo los caminos.

TAD VERTEX

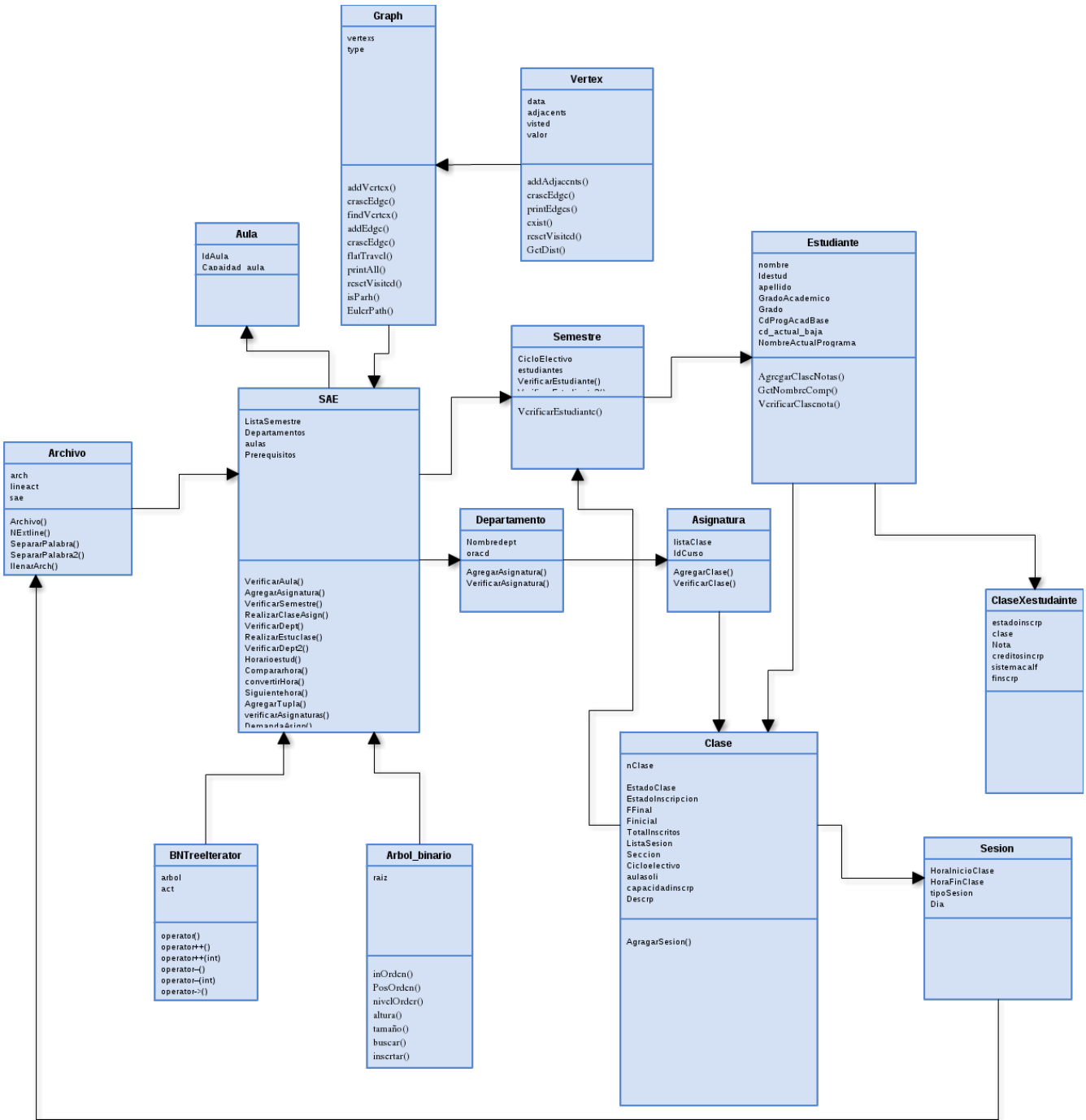
Conjunto mínimo de datos.

- data<plantilla>es el dato que se le asigna al vertice.
- adjacents<multi mapa>es donde se guardan los vertices que se conectan con los vertices.}
- visited<boleano> este atributo permite saber si ya se visto el vertice.
- valor<numerico> es el valor que se le da para el recorrido de dijkstra.

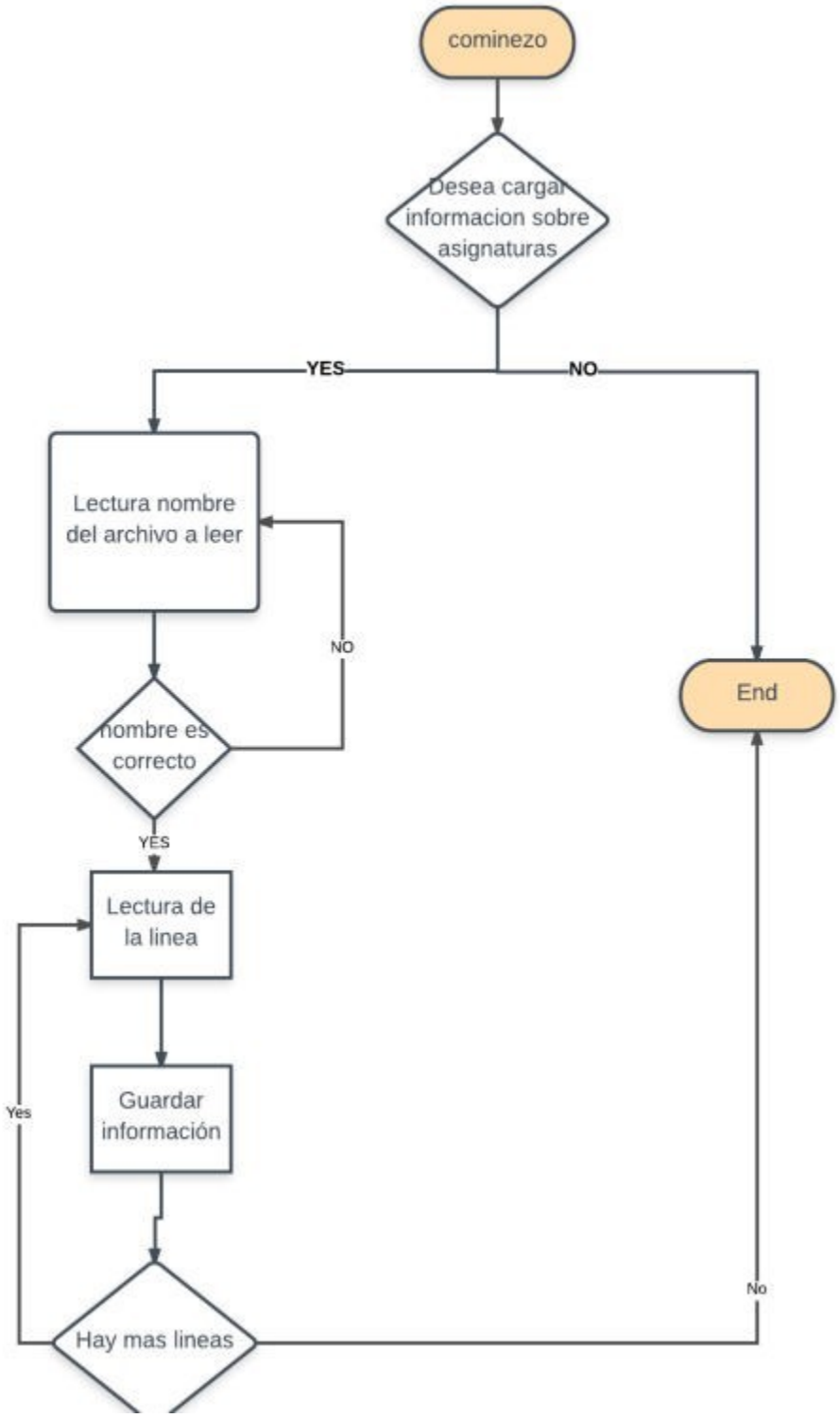
Comportamiento (Operaciones) del objeto

- addAjacents<addAjacents()>agrega un vertice al multi mapa de adyacencia.
- eraseEdge<eraseEdge()>Elimina un camino que tenga este vertice.
- PrintEdges<PrintEdges()>imprime todos los caminos de este vertice.
- exist<exist()>verifica si el vertice existe.
- resetVisited<resetVisited()> modifica el atributo visted del vertice.
- GetDist<GetDist()>retorna el peso del camino.
- ExistEdge<ExistEdge()>verifica si un camino existe en el vertice.

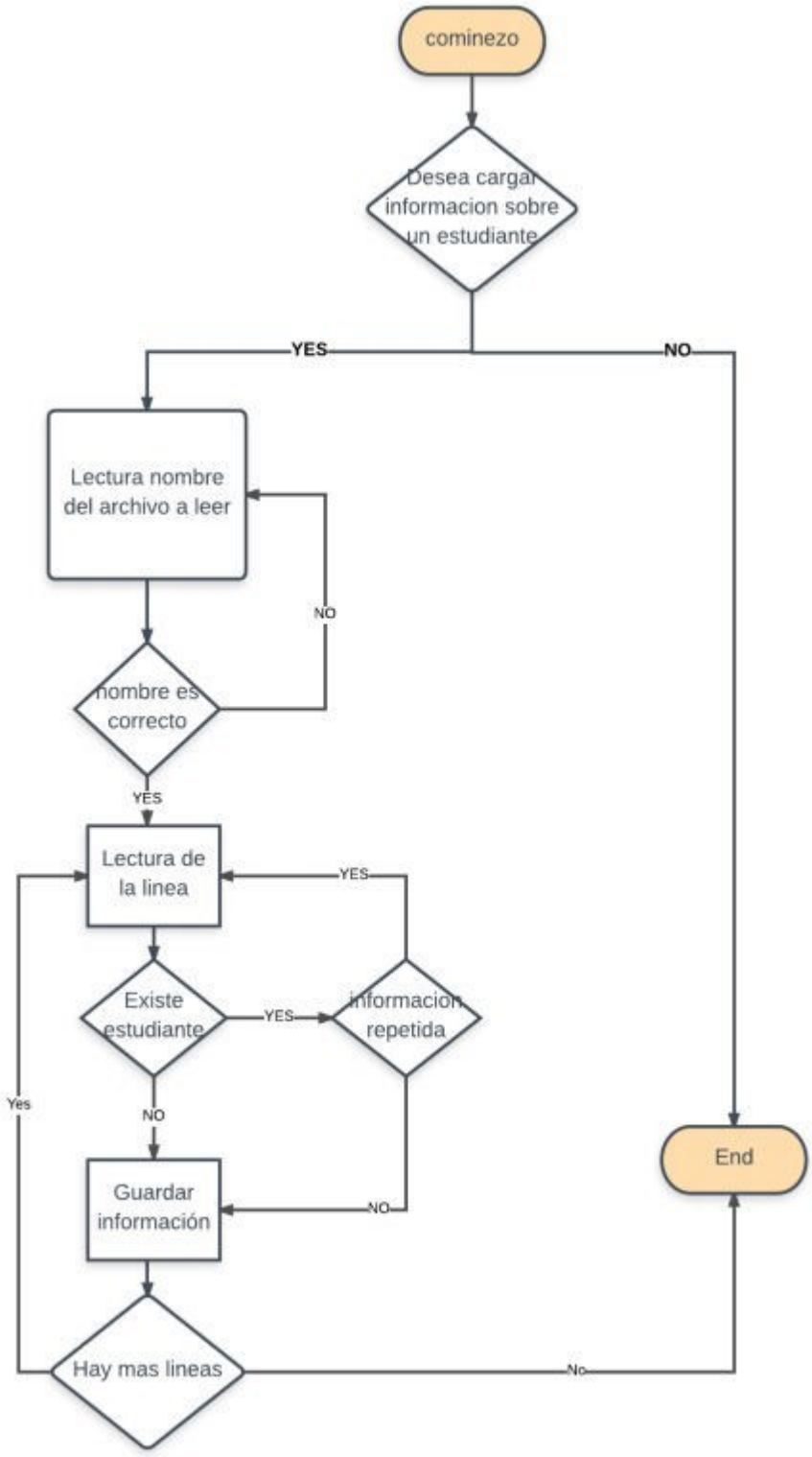
Diagrama de relación entre TADs



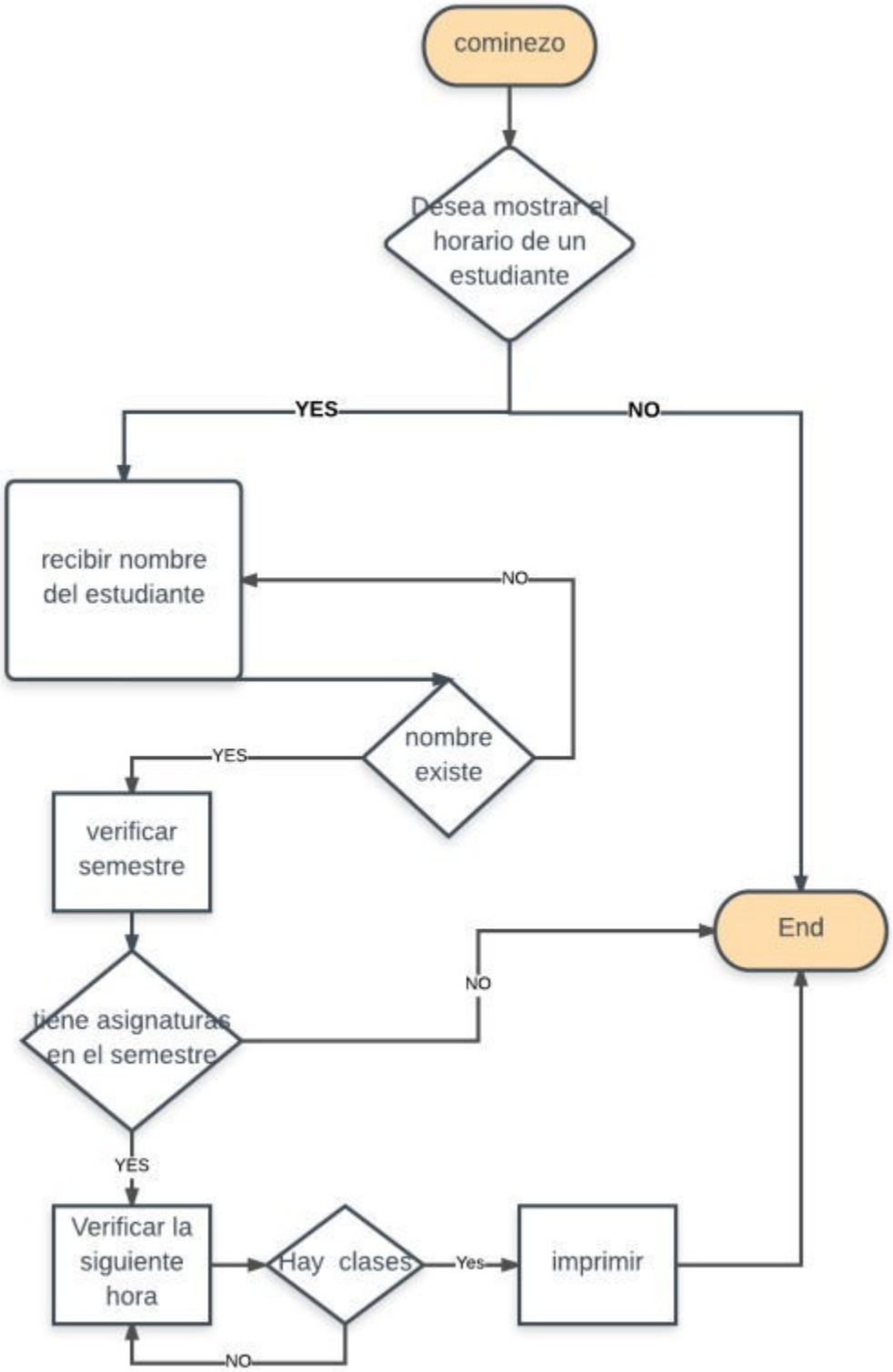
ENTREGA 1
PUNTO 1:Cargar información de la asigntura



PUNTO 2: cargar informacion de los estudaintes

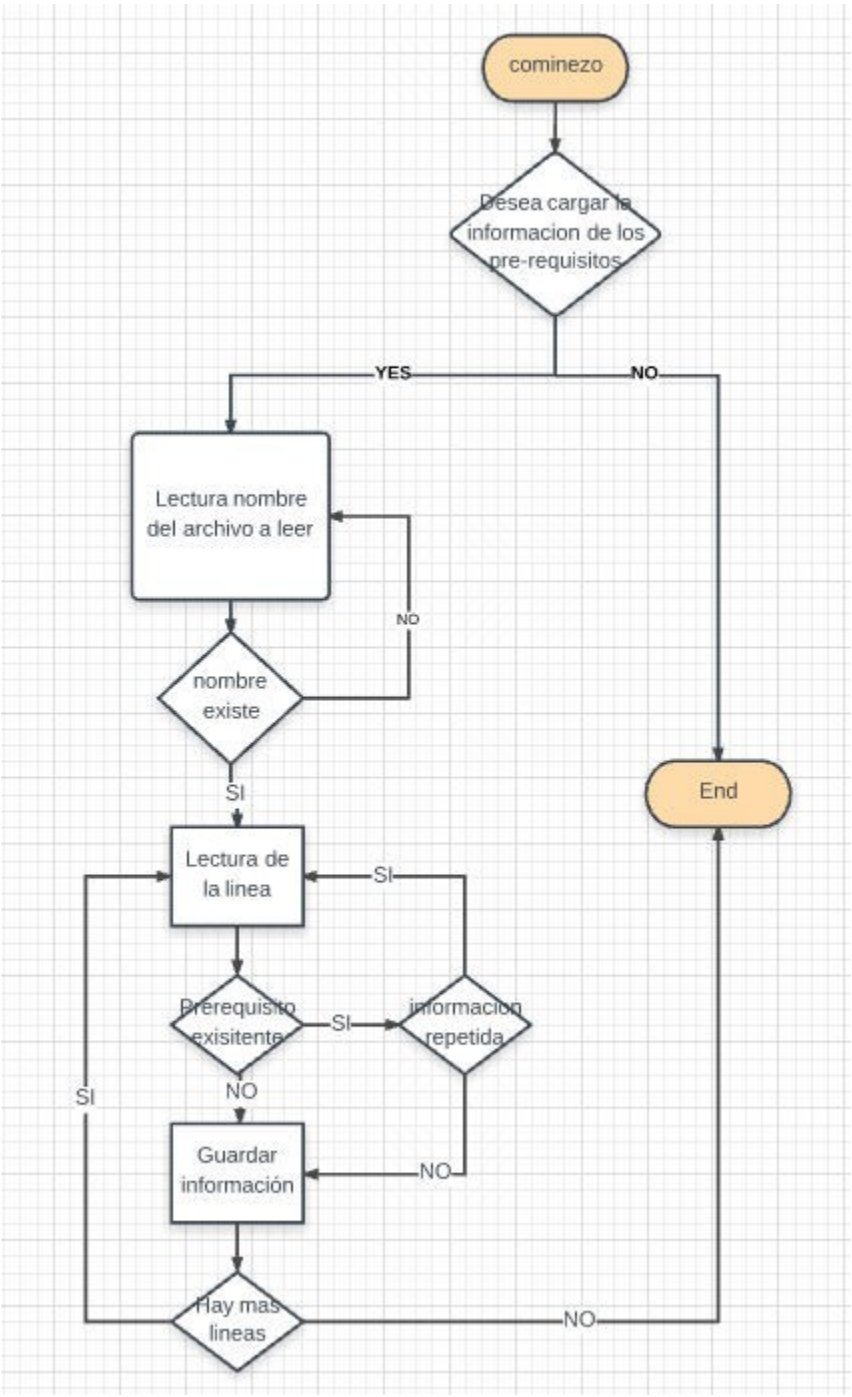


PUNTO 3: mostrar el horario de un estudiante

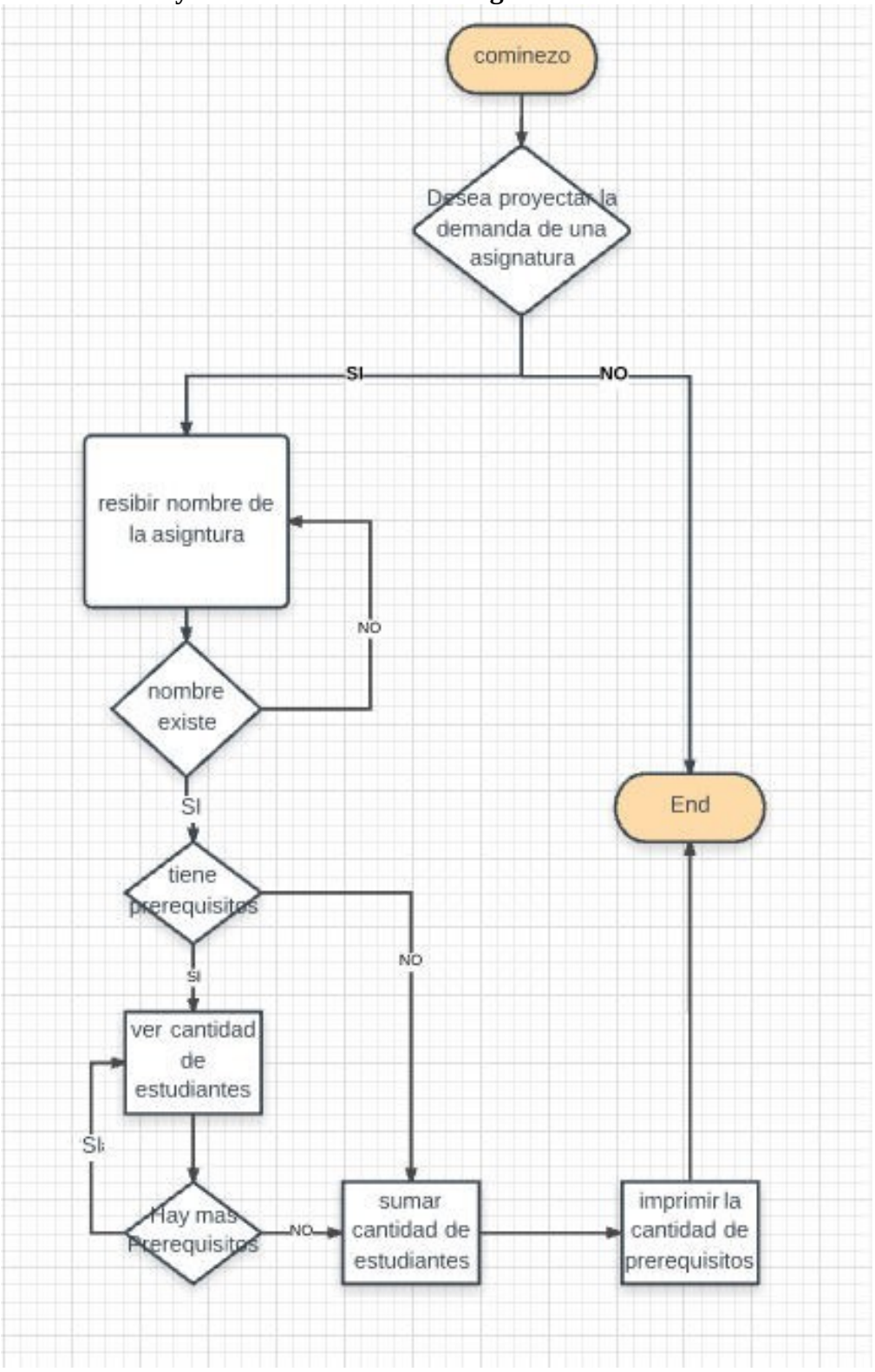


ENTREGA 2

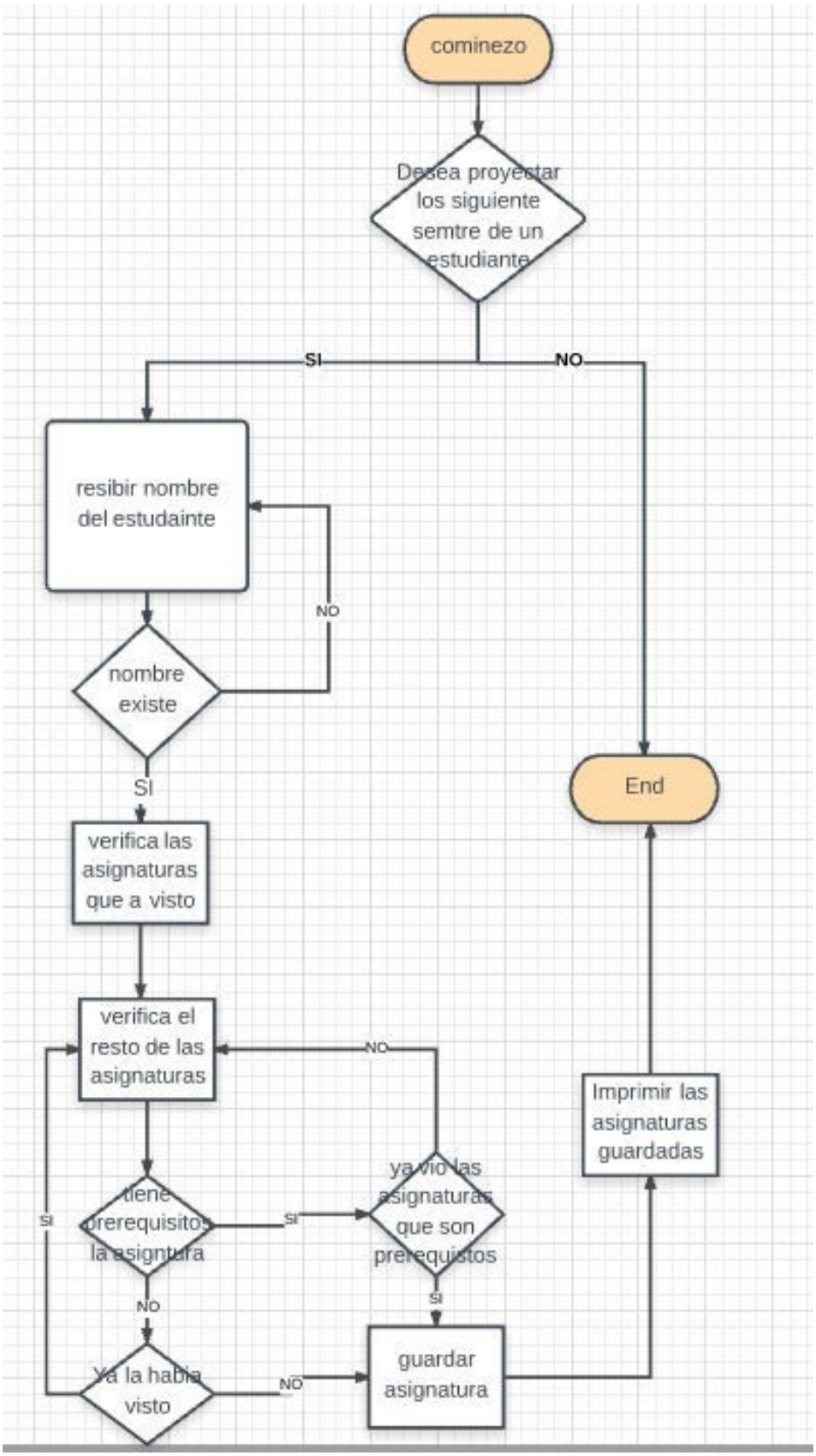
PUNTO 1: Cargar informacion de los prerequisitos



PUNTO 2: Proyectar demanda de una asignatura



PUNTO 3: Proyeccion de semestre por estudiante



PUNTO 4:Generar red social