



Centro Universitario de Ciencias Exactas e Ingenierías

Bases de Datos Avanzadas

Maestra: Murillo Leño María Magdalena

Alumnos: Díaz Márquez Oscar/Garfía Pahua José

Sección: D02

Proyecto

Minimundo

Se requiere realizar un software para la Escuela de Iniciación Artística (INBA) que lleve la relación de los registros de los alumnos y profesores.

La escuela divide a los alumnos en grupos de primero a sexto semestre en los cuales existen dos categorías: infantil (6 a 13 años) y juvenil (14 a 21 años).

La escuela cuenta con las siguientes disciplinas: Danza, Teatro, Música y Artes plásticas, también la escuela tiene la oportunidad de que los alumnos puedan asistir semanalmente o curso sabatino intensivo.

El primer semestre los alumnos tanto de la categoría infantil y juvenil tienen clases con un tronco común. A partir del segundo semestre y hasta el fin del curso (el cual tiene la duración de tres años), se decide por una especialidad.

Los datos a capturar de los alumnos son: matrícula, nombre, fecha de nacimiento, categoría, edad, correo electrónico, teléfono, nombre del tutor, especialidad elegida y curp.

Los datos a capturar de los profesores son: matrícula, nombre, grado académico, edad, correo electrónico, teléfono, antigüedad, sueldo, curp y tipo de profesor (titular o suplente).

Entrevista

Nombre del entrevistado: José Manuel Hernández.

Nombre de la profesión y cargo que realiza: Su cargo es ser el encargado de Control Escolar.

- Domicilio: Calle Degollado #20 entre Morelos y Pedro Moreno, zona centro Guadalajara Jalisco.
- Teléfono: 36583931
- Correo electrónico: artesproculata@hotmail.com

1.- ¿Cuál es el objetivo principal que quieres que realice el software?

R= Que lleve una relación de los registros de los alumnos de acuerdo a su enfoque artístico junto con sus materias y profesores.

2.- ¿Cuántos tipos de usuarios deberá tener el sistema?

R= Dos usuarios, hay dos puestos principales: Director y Control Escolar.

3.- Y los 2 usuarios tienen acceso a toda la información del sistema, o hay alguno que tenga acceso a diferente información o que pueda hacer cosas que otros no.

Los 2 tienen acceso a todo, pero solo el de control escolar puede hacer modificaciones a datos de los alumnos.

4.- ¿Tienes conocimiento alguno acerca de computación o del manejo de una computadora?

R= Sí, ya con anterioridad he manejado sistemas diferentes, y se más de lo básico acerca de computación.

5.- ¿Cuál sería la principal ayuda o beneficio que tendrías usando este sistema?

R= Pues, me sería de gran ayuda para llevar de una mejor manera los registros de cada uno de los alumnos, y además de que así estarían mejor guardados y la pérdida de información sería nula.

6.- ¿Además de que el sistema, guarde los registros de los alumnos, deberá realizar de igual manera algo con otro tipo de registros, como de profesores por ejemplo?

R= Sí, así como tendrá una relación de los registros de los alumnos, deberá de hacerlo respectivamente con los profesores.

7.- ¿Cuáles serían los distintos enfoques o disciplinas artísticas que pueden llevar o escoger los alumnos?

R= La escuela se dedica a impartir 4 clases de disciplinas artísticas como lo son: clases de música, de danza, de teatro y de artes plásticas, y los alumnos pueden escoger una de esas 4.

8.- ¿Hay diferentes tipo de alumnos, me refiero a que si hay alumnos que sean becados o de intercambio, o algo parecido?

R= Si, existen dos categorías: Infantil (6 a 13 años), Juvenil (14 a 21 años), y en cuanto a la beca, la escuela maneja dos tipos de cuotas, ya sea completa o becada.

9.- ¿Cuántos grupos tiene la escuela?

R= La escuela se divide en grupos de primer a sexto semestre.

10.- ¿Hay diferentes tipos de cursos a la que los alumnos puedan ir a clases o se maneja uno solo?

R= Para ambas categorías existe la opción de ir semanalmente o ir a un curso sabatino intensivo.

11.- ¿Cuáles serían los datos a capturar para los alumnos?

R= Se requiere: matrícula, nombre, fecha de nacimiento, categoría, edad, correo electrónico, teléfono, nombre del tutor, y que documentos tiene registrados en su expediente.

12.- ¿De qué documentos está conformado el expediente de los alumnos?

R= Contiene su ficha de inscripción, cuatro fotografías, acta de nacimiento, y curp.

13.- ¿Estos documentos serán guardados en la base de datos o solo se registra que se entregaron?

R= Solo se registra que entregaron documentos.

14.- ¿Cuáles serían los datos a capturar para los profesores?

R= Se requiere: matrícula, nombre, grado académico, edad, correo electrónico, teléfono y si es maestro titular o suplente.

15.- ¿Será necesario registrar materias? En caso de necesitarlo ¿Qué información se necesita guardar de cada materia?

R= No, solo al segundo semestre, es necesario decidirse por una especialidad.

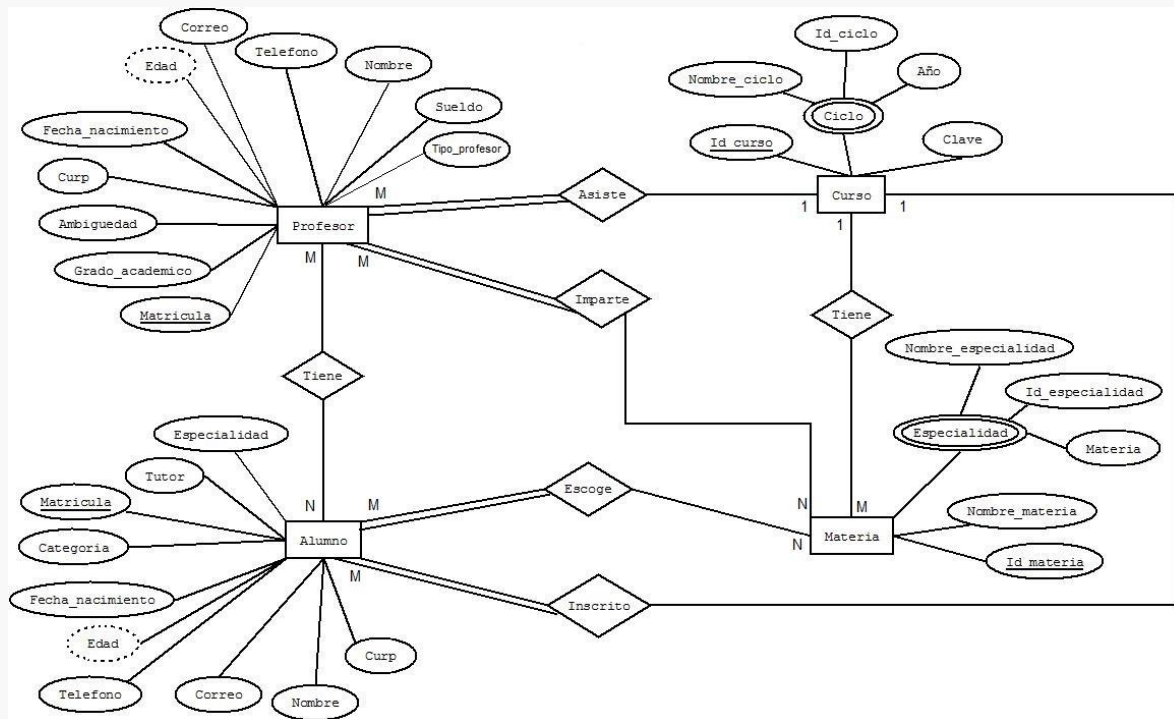
Objetivo

El objetivo principal del sistema es mantener un registro centralizado y ordenado de la información académica de los alumnos del instituto. En especial para poder conocer de manera rápida los detalles de la especialidad que eligió cada alumno.

Como objetivos secundarios del sistema se encuentran:

- Mantener un registro académico adecuado de los profesores y sus diversas asignaturas que imparten.
- Proporcionar una herramienta de consulta de información de ciclos escolares anteriores.

Diagrama ER

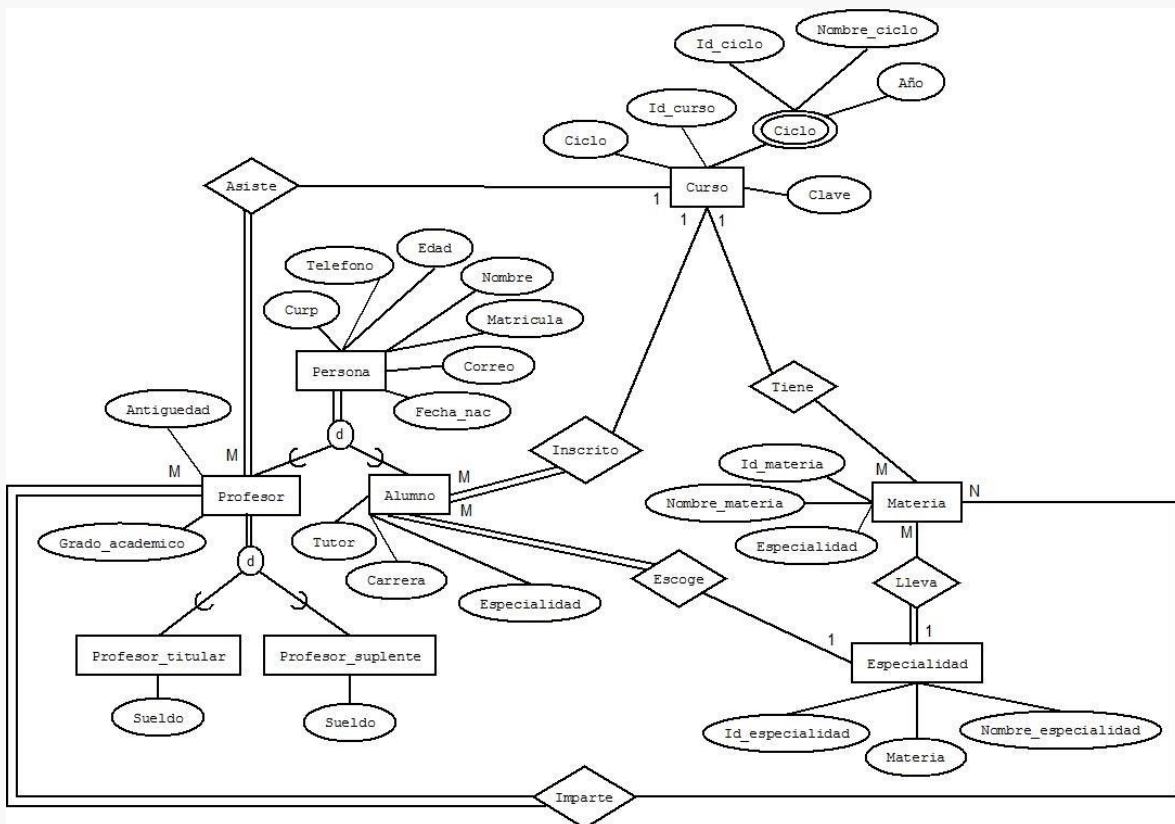


Diccionario de Datos

Tabla (entidad)	Campo (atributo)	Tipo de Dato y Longitud	Observaciones
Profesor	Nom_profesor	Varchar(30)	no nulo
	Matrícula	Varchar(20)	Llave primaria
	Curp	Varchar(30)	No nulo
	Correo	Varchar(30)	No nulo
	Fecha_nacimiento	Varchar(15)	No nulo
	Telefono	integer	No nulo
	Edad	integer	No nulo
	Grado_academico	Varchar(20)	No nulo
	Antigüedad	Varchar(15)	No nulo
	Sueldo	float	No nulo
	Tipo_profesor	Varchar(30)	No nulo
Alumno	Nom_alumno	Varchar(30)	no nulo
	Matrícula	Varchar(20)	Llave primaria
	Curp	Varchar(30)	No nulo
	Correo	Varchar(30)	No nulo
	Fecha_nacimiento	Varchar(15)	No nulo
	Telefono	integer	No nulo
	Edad	integer	No nulo
	Especialidad	Varchar(30)	No nulo

	Tutor	Varchar(30)	No nulo
	Categoría	Varchar(30)	No nulo
Materia	Id_materia	integer	Llave primaria
	Nom_materia	Varchar(30)	No nulo
	Id_especialidad	integer	Llave foranea
Especialidad	Id_especialidad	integer	Llave primaria
	Nom_especialidad	Varchar(30)	No nulo
	Nomb_materia	Varchar(30)	Llave foranea
Curso	Id_curso	integer	Llave primaria
	clave	Varchar(15)	No nulo
	Id_ciclo	integer	No nulo
	Ciclo	Varchar(10)	No nulo
	Nom_ciclo	Varchar(30)	No nulo
	Año	integer	No nulo

Diagrama EER



Modelo Relacional

Persona						
Matrícula	Nombre	Edad	Correo	Teléfono	Curp	Fecha_nacimiento

Profesor										
Matrícula	Nombre	Edad	Correo	Teléfono	Curp	Fech_nac	Antigüedad	Grado	Tipo	Id_curso

Alumno												
Matrícula	Nombre	Edad	Correo	Teléfono	Curp	Fech_nac	Tutor	Carrera	Espec.	Id_curso	Id_espec.	

Profesor_titular		
Antigüedad	Grado_académico	Sueldo

Profesor_suplente		
Antigüedad	Grado_académico	Sueldo

Materia				
Id_materia	Nombre_materia	Especialidad	Id_especialidad	Id_curso

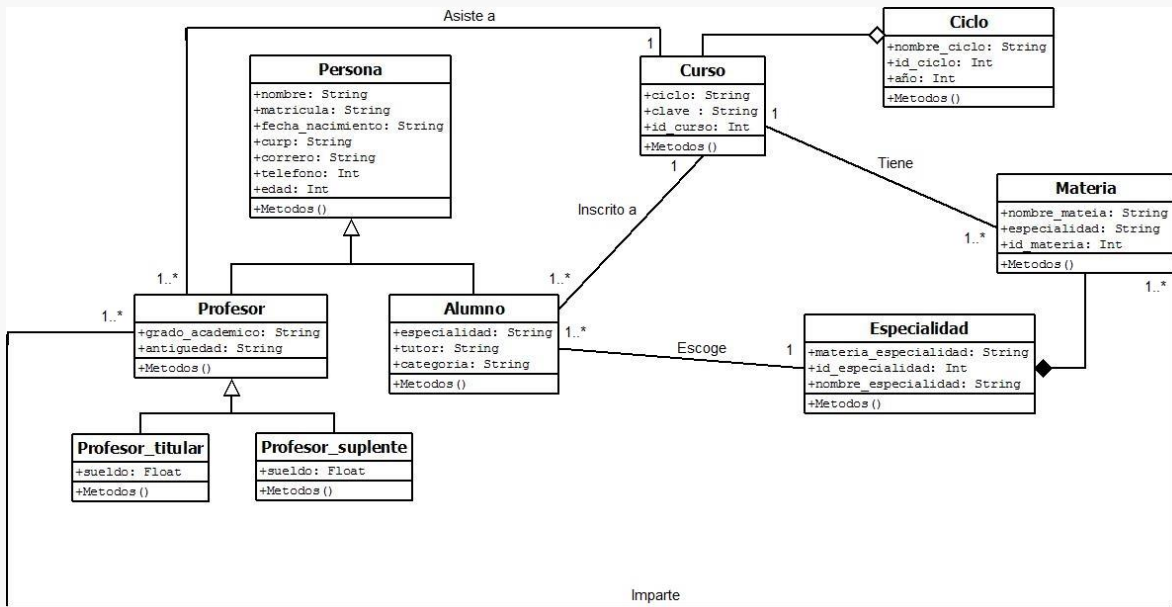
Especialidad		
Id_especialidad	Nombre_especialidad	Materia

Curso		
Id_curso	Ciclo	Clave

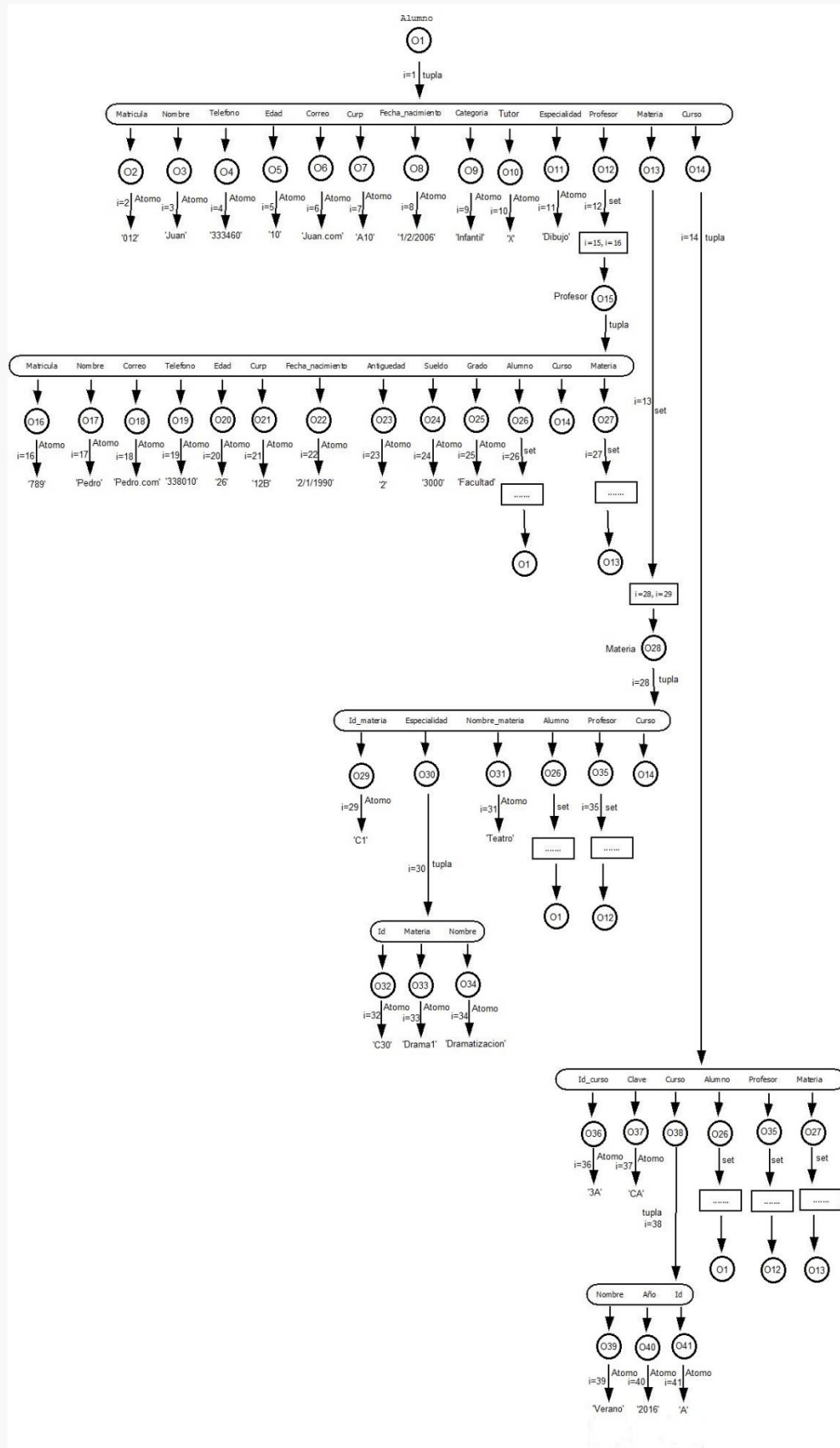
Ciclo_curso			
Id_curso	Id_ciclo	Nombre_ciclo	Año

Imparte	
Matrícula	Id_materia

Diagrama de Clases



Grafo

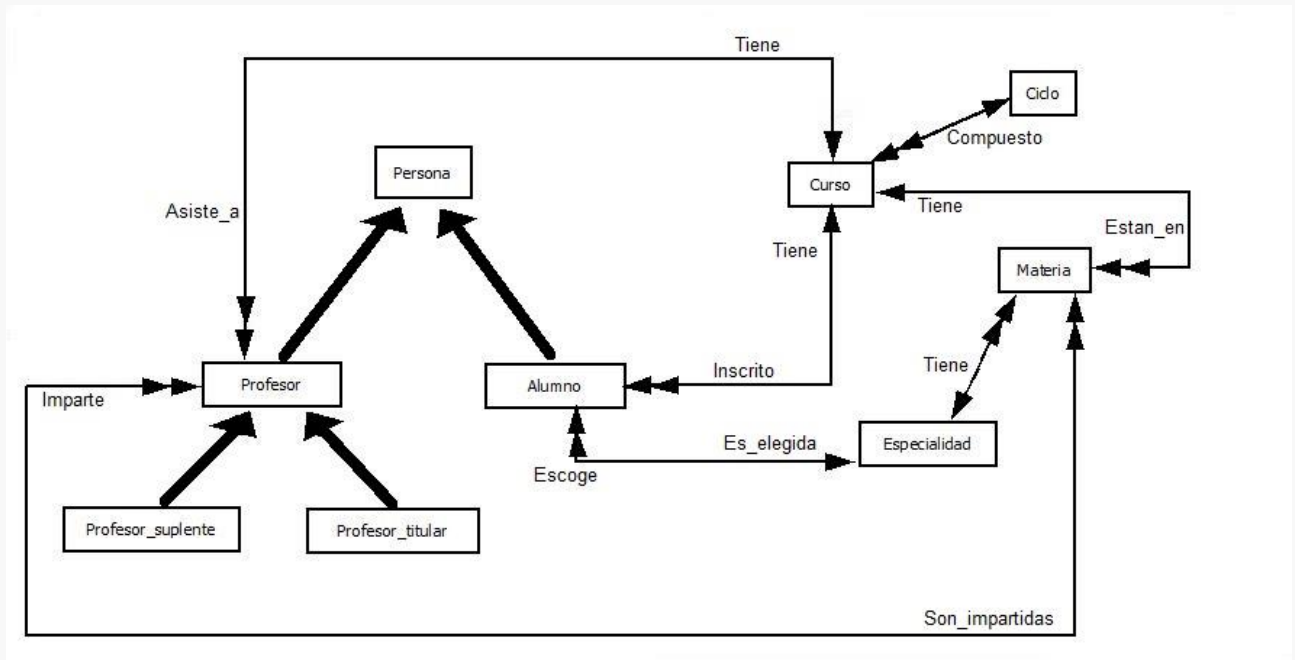


Constructor

- O1 = (i1, tupla, <matricula: i2, nombre: i3, telefono: i4, edad: i5, correo: i6, curp: i7, fecha_nacimiento: i8, tutor: i9, categoría: i10, especialidad: i11, profesor: i12, materia: i13, curso: i14>);
- O2 = (i2, atomo, '012');
- O3 = (i3, atomo, 'Juan');
- O4 = (i4, atomo, '333460');
- O5 = (i5, atomo, '10');
- O6 = (i6, atomo, 'Juan.com');
- O7 = (i7, atomo, 'A10');
- O8 = (i8, atomo, '1/2/2006');
- O9 = (i9, atomo, 'X');
- O10 = (i10, atomo, 'Infantil');
- O11 = (i11, atomo, 'Dibujo');
- O12 = (i12, set, {i15, i16});
- O13 = (i13, set, {i28, i29});
- O14 = (i14, tupla, <Id_curso: i36, clave: i37, ciclo: i38, alumno: i26, profesor: i36, materia: i28>);
- O15 = (i15, tupla, <matricula: i16, nombre: i17, correo: i18, teléfono: i19, edad: i20, curp: i21, fecha_nacimiento: i22, antigüedad: i23, sueldo: i24, grado: i25, alumno: i26, curso: i14, materia: i27>);
- O16 = (i16, atomo, '789');
- O17 = (i17, atomo, 'Pedro');
- O18 = (i18, atomo, 'Pedro.com');
- O19 = (i19, atomo, '338010');
- O20 = (i20, atomo, '26');
- O21 = (i21, atomo, '12B');
- O22 = (i22, atomo, '2/1/1990');
- O23 = (i23, atomo, '2');
- O24 = (i24, atomo, '3000');
- O25 = (i25, atomo, 'Facultad');
- O26 = (i26, set, {i1});
- O27 = (i27, set, {i13});
- O28 = (i28, tupla, <Id_materia: i29, especialidad: i30, nombre_materia: i31, alumno: i26, profesor: i35, curso: i14>);
- O29 = (i29, atomo, 'C1');
- O30 = (i30, tupla, <Id: i32, materia: i33, nombre: i34>);
- O31 = (i31, atomo, 'Teatro');
- O32 = (i32, atomo, 'C30');
- O33 = (i33, atomo, 'Drama1');
- O34 = (i34, atomo, 'Dramatizacion');

- O35 = (i35, set, {i12});
- O36 = (i36, atomo, '3A');
- O37 = (i37, atomo, 'CA');
- O38 = (i38, tupla, <nombre: i39, año: i40, id: i41>);
- O39 = (i39, atomo, 'Verano');
- O40 = (i40, atomo, '2016');
- O41 = (i41, atomo, 'A');

Diagrama ODL



Definición los tipos de objetos y sus operaciones del esquema de la BD mediante el Lenguaje de Programación Orientada a Objetos "Java".

```

public class EscArte {
    public EscArte() {
    }

    public static void main(String[] args) {
        // Main
    }
}

//escuela de arte
class Persona {

```

```

//atributos
String Nombre, Matricula, FechaNac, Curp, Correo;
int Telefono, Edad;

//constructor
public Persona(String Nombre, String Matricula, String FechaNac, String
Curp,
                String Correo, int Telefono, int Edad) {
    this.Nombre = Nombre;
    this.Matricula = Matricula;
    this.FechaNac = FechaNac;
    this.Curp = Curp;
    this.Corrreo = Correo;
    this.Telefono = Telefono;
    this.Edad = Edad;
}

//constructor
//metodos regresar
public String RegNombre(){
    return Nombre;
}
public String RegMatricula(){
    return Matricula;
}
public String RegFechaNac(){
    return FechaNac;
}
public String RegCurp(){
    return Curp;
}
public String RegCorreo(){
    return Correo;
}

```

```
}  
public int RegTelefono(){  
    return Telefono;  
}  
public int RegEdad(){  
    return Edad;  
}  
//metodos modificar  
public void ModNombre (String n){  
    Nombre = n;  
}  
public void ModMatricula (String m ){  
    Matricula = m;  
}  
public void ModFechaNac (String f){  
    FechaNac = f;  
}  
public void ModCurp (String c){  
    Curp = c;  
}  
public void ModCorreo (String co){  
    Correo = co;  
}  
public void ModTelefono (int t){  
    Telefono = t;  
}  
public void ModEdad(int e){  
    Edad = e;  
}  
} //persona
```

```

class Profesor extends Persona {
    //atributos
    String GradoAcad, Antiguedad, Materia;
    float Sueldo;
    Boolean Suplente = false;

    //constructor
    public Profesor(String Nombre, String Matricula, String FechaNac, String
    Curp,
        String Correo, int Telefono, int Edad, String GradoAcad, String
    Antiguedad,
        String Materia, float Sueldo, Boolean Suplente){
        super(Nombre, Matricula, FechaNac, Curp, Correo, Telefono, Edad);
        this.GradoAcad = GradoAcad;
        this.Antiguedad = Antiguedad;
        this.Materia = Materia;
        this.Suplente = Suplente;
    }//constructor

    //metodos regresar
    public String RegGradoAcad(){
        return GradoAcad;
    }
    public String RegAntiguedad(){
        return Antiguedad;
    }
    public String RegMateria(){
        return Materia;
    }
    public float RegSueldo(){
        return Sueldo;
    }

```

```

    }
    public Boolean RegSuplente(){
        return Suplente;
    }
    //metodos modificar
    public void ModGradoAcad (String g){
        GradoAcad = g;
    }
    public void ModAntiguedad (String a){
        Antiguedad = a;
    }
    public void ModMateria (String ma){
        Materia = ma;
    }
    public void ModSueldo (float s){
        Sueldo = s;
    }
    public void ModSuplente (Boolean su){
        Suplente = su;
    }
} //profesor
class Alumno extends Persona {
    //atributos
    String Especialidad, Tutor, Categoria;

    //constructor
    public Alumno(String Nombre, String Matricula, String FechaNac, String
    Curp,
        String Correo, int Telefono, int Edad, String Especialidad,
        String Tutor, String Categoria){
        super(Nombre, Matricula, FechaNac, Curp, Correo, Telefono, Edad);
    }
}

```

```

        this.Especialidad = Especialidad;
        this.Tutor = Tutor;
        this.Categoria = Categoria;
    } //constructor

//metodos regresar
public String RegEspecialidad(){
    return Especialidad;
}
public String Regtutor(){
    return Tutor;
}
public String RegCategoria(){
    return Categoria;
}

//metodos modificar
public void ModEspecialidad(String es){
    Especialidad = es;
}

public void ModTutor(String tu){
    Tutor = tu;
}
public void ModCategoria(String ca){
    Categoria = ca;
}
} //Alumno
class Curso{
    //atributos
    String CicloCurso, MateriaCurso, profesorCurso, alumnoCurso, Clave;
    int id_Curso;

```



```

        //constructor
        public Curso(String CicloCurso, String MateriaCurso, String
alumnoCurso, String Clave,
                String profesorCurso, int id_Curso){
            this.CicloCurso      = CicloCurso;
            this.MateriaCurso    = MateriaCurso;
            this.profesorCurso    = profesorCurso;
            this.alumnoCurso     = alumnoCurso;
            this.Clave           = Clave;
            this.id_Curso        = id_Curso;
        }
        //metodos regresar
        public String RegCicloCurso(){
            return CicloCurso;
        }
        public String RegMateriaCurso(){
            return MateriaCurso;
        }
        public String RegProfesorCurso(){
            return profesorCurso;
        }
        public String RegAlumnoCurso(){
            return alumnoCurso;
        }
        public String RegClave(){
            return Clave;
        }
        public int RegId_Curso(){
            return id_Curso;
        }
        //metodos modificar

```

```

public void ModCicloCurso(String ci){
    CicloCurso = ci;
}

public void ModMateriaCurso(String mat){
    MateriaCurso = mat;
}

public void ModProfesorCurso(String pro){
    profesorCurso = pro;
}

public void ModAlumnoCurso(String alun){
    alumnoCurso = alun;
}

public void ModClave(String cla){
    Clave = cla;
}

public void Mod (int icu){
    id_Curso = icu;
}

} //Curso

class Ciclo{
    //atributos
    String NombreCiclo;
    int id_Ciclo, anhio;

    //constructor
    public Ciclo(String NombreCiclo, int id_Ciclo, int anhio){
        this.NombreCiclo = NombreCiclo;
        this.id_Ciclo = id_Ciclo;
        this.anhio = anhio;
    }
}

```

```

    }

    //metodos regresar
    public String RegNombreCiclo(){
        return NombreCiclo;
    }

    public int RegidCiclo(){
        return id_Ciclo;
    }

    public int Reganhio(){
        return anhio;
    }

    //metodos Modificar
    public void ModNombreCiclo(String nci){
        NombreCiclo = nci;
    }

    public void ModidCiclo(int idci ){
        id_Ciclo = idci;
    }

    public void Modanhio(int anh){
        anhio = anh;
    }

} //ciclo

class MateriaEsc{
    //atributos
    String NombreMateria, EspecialidadMateria;
    int id_materia;

    //constructor
    public MateriaEsc(String NombreMateria,String EspecialidadMateria, int
id_materia){

```

```

        this.NombreMateria      = NombreMateria;
        this.EspecialidadMateria = EspecialidadMateria;
        this.id_materia         = id_materia;
    }

//metodos regresar
public String RegNombreMateria(){
    return NombreMateria;
}

public String RegEspecialidadMateria(){
    return EspecialidadMateria;
}

public int RegIdMateria(){
    return id_materia;
}

//metodos modificar
public void ModNombreMateria(String nombM){
    NombreMateria = nombM;
}

public void ModEspecialidadMateria(String espmate){
    EspecialidadMateria = espmate;
}

public void ModIdMateria(int idma){
    id_materia = idma;
}

}

//materiaesc

class Especialidad{
    //atributos
    String NombreEspecialidad, MateriaEspecializante;
    int id_Especialidad;
}

```

```

        //constructor
        public Especialidad(String NombreEspecialidad, String
MateriEspecializante, int id_Especialidad){
            this.NombreEspecialidad    = NombreEspecialidad;
            this.MateriaEspecializante = MateriaEspecializante;
            this.id_Especialidad        = id_Especialidad;
        }
//metodos regresar
public String RegNombreEspecialidad(){
    return NombreEspecialidad;
}
public String RegMateriEspecializante(){
    return MateriaEspecializante;
}
public int RegIdEspecialidad(){
    return id_Especialidad;
}
//metodos Modificar
public void ModnombreEspecialidad(String nombesp){
    NombreEspecialidad = nombesp;
}

public void ModMateriaEspecializante(String matesp){
    MateriaEspecializante = matesp;
}
public void Mod(int idesp){
    id_Especialidad = idesp;
}
} //especialidad

```

Consultas OQL.

1.- Obtener el total de alumnos que llevan la materia de dibujo.

* Count (Alumno)

For All e In Alumno: e.materia = 'Dibujo';

2.- Obtener el total de profesores que tienen salario mayor a 3000.

* Count (Profesor)

For All Profesor: e.salario > 3000;

3.- Obtener las edades de todos los alumnos llamados Pedro.

* Select distinct x.edad From x in Alumno Where x.nombre = 'Pedro';

4.- Obtener la edad y salario de todas las maestras llamadas María.

* Select distinct struct (e:x.edad, s:x.salario)

From x in Profesor Where x.nombre = 'María';

PROLOG

% alumno(matricula, nombre, fechaN, curp, correo, telef, edad, especialidad, tutor, categoria).

:- dynamic(alumno/2).

alumno(hv333, [jose, junio1893, zasbt, beast@, ext123, veinte, actor, maria, grado]).

% maestro(matricula, nombre, fechaN, curp, correo, telef, edad, gradoA, antig, materia).

:- dynamic(maestro/2).

maestro(hs321, [andres, agosto1284, asdf, corp@, ext3456, treinta, profesor, doceanhios, actuacion]).

%mostramos todos los alumnos y maestros

listar :- setof(X, (P,H)^(alumno(P,H), member(X,H)), L), write(L),nl,

setof(Y, (Q,R)^(maestro(Q,R), member(Y,R)), M), write(M),nl..

%agregamos un alumno a la base de datos

```
adicionarA :- write('Matricula '), read(Matricula),  
              write('Escriba 9 por favor '),  
              read(Cant), crearAlum(Cant, A),  
              Alum = alumno(Matricula,A),  
              asserta(Alum).
```

```
crearAlum(0,[]) :- !,true.
```

```
crearAlum(C, [DATO|L]). :- read(DATO), Ctemp is C - 1, crearAlum(Ctemp,L).
```

%agregamos un maestro a la base de datos

```
adicionarM :- write('Matricula '), read(Matricula),  
              write('Escriba 9 por favor '),  
              read(Cant), crearMaest(Cant, A),  
              Alum = alumno(Matricula,A),  
              asserta(Alum).
```

```
crearMaest(0,[]) :- !,true.
```

```
crearMaest(C, [DATO|L]). :- read(DATO), Ctemp is C - 1, crearMaest(Ctemp,L).
```

%eliminamos cualquier registro que coincida

```
eliminar :- write('Matricula '), read(Matricula).  
           retract(alumno(Matricula,_)), !.
```

```
eliminar :- write('No existe tal persona \n').
```

%borramos todos los registros

```
borrar :- abolish(alumno/2), abolish(maestro/2),  
         write('Se han borrado todos los datos /n').
```

%menu que aparece en consola

```
menu :- write('***** MENU ESCUELA INBA *****'),nl,  
        write('1. AGREGAR ALUMNO. '),nl,
```

```

write('2. AGREGAR MAESTRO. '),nl,
write('3. ELIMINAR MAESTRO O ALUMNO. '),nl,
write('4. BORRAR TODOS LOS REGISTROS. '),nl,
write('5. LISTAR TODOS LOS REGISTROS. '),nl,
write('*** PRESIONA CERO PARA SALIR ***'),nl,
write('*** ELIGE UNA OPCION ***'), read(Opcion),
ejecutar(Opcion).

```

%dependiendo la opcion se ejecuta el menu

ejecutar(Opcion):- Opcion == 1, adicionarA , menu;

Opcion == 2, adicionarM , menu;

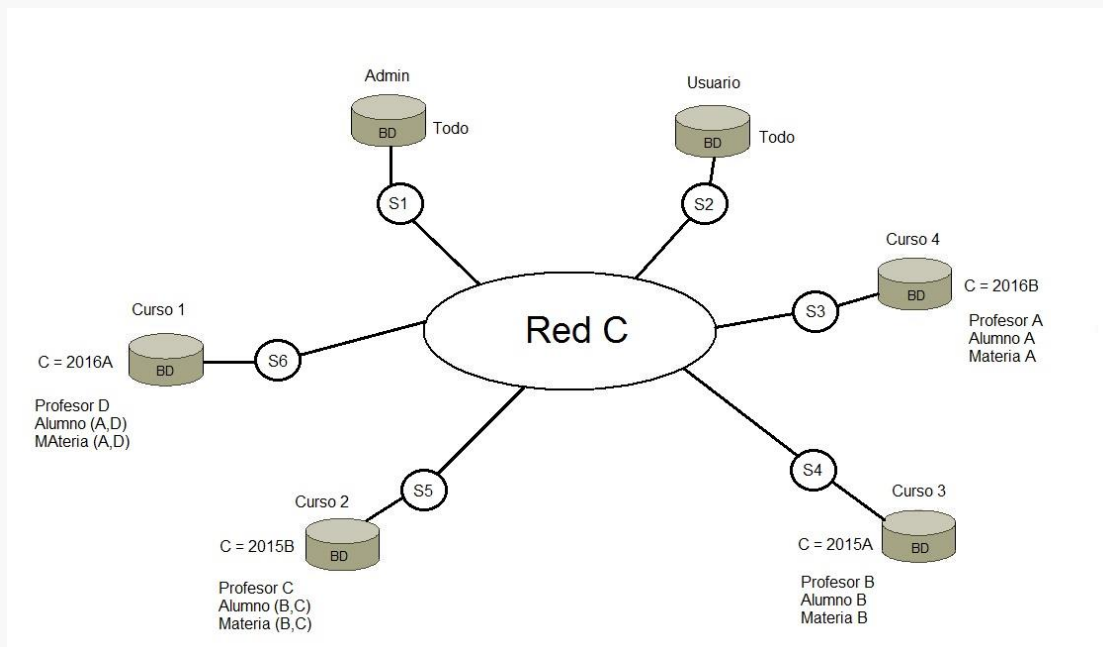
Opcion == 3, eliminar , menu;

Opcion == 4, borrar , menu;

Opcion == 5, listar , menu;

Opcion == 0, true.

Diagrama Distribuido



Fragmentaciones

- **Fragmentación Horizontal**

1.- Mostrar a los alumnos que se encuentran en la categoría Juvenil.

Select *from Alumno where categoría = 'Juvenil';
 σ categoría = 'Juvenil' (Alumno)

2.- Mostrar a los profesores que tienen sueldo de 3000.

Select *from Profesor where sueldo = 3000;
 σ categoría = 3000 (Profesor)

- **Fragmentación Vertical**

1.- Mostrar el Id y la clave de los cursos.

Select Id_curso, clave from Curso;
 π Id_curso, clave (Curso)

2.- Mostrar el Id y el nombre de las materias.

Select Id_materia, nombre_materia from Materia;
 π Id_materia, nombre_materia (Materia)

- **Fragmentación Mixta**

1.- Mostrar el nombre de los alumnos y de sus tutores, de aquellos que escogieron la especialidad de dibujo.

Select nombre, tutor from Alumno where especialidad = 'Dibujo';
 π nombre, tutor (σ especialidad = 'Dibujo' (Alumno))

2.- Mostrar el nombre y la edad de los profesores que tienen grado académico de universidad.

Select nombre, edad from Profesor where grado_academico = 'Universidad';
 π nombre, edad (σ grado_academico = 'Universidad' (Profesor))

- **Fragmentación Horizontal Derivada**

1.- Mostrar los cursos que su ID sea igual a 2.

Select *from Curso where Id_curso = 2;
 σ Id_curso = 2 (Curso)

Mostrar los profesores que asisten a los cursos cuyo ID es igual a 2.

Select *from Profesor where Id_curso = 2;
 σ Id_curso = 2 (Profesor)

Mostrar las materias que se imparten en los cursos cuyo ID es igual a 2.

Select *from Materia where Id_curso = 2;

$\sigma \text{ Id_curso} = 2 \text{ (Materia)}$

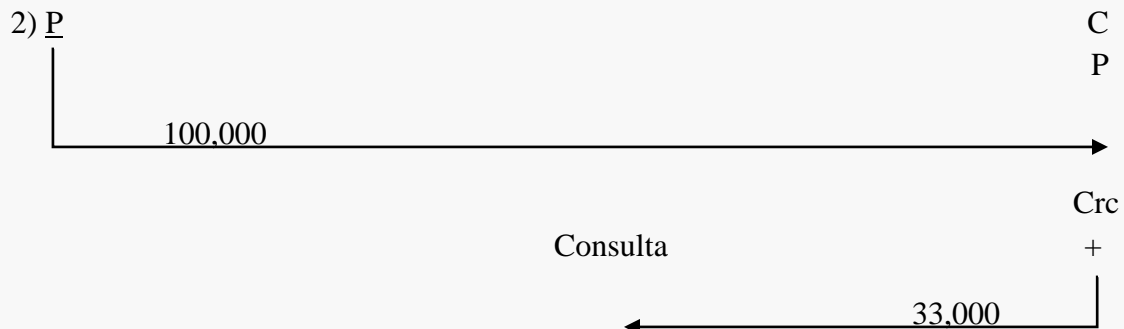
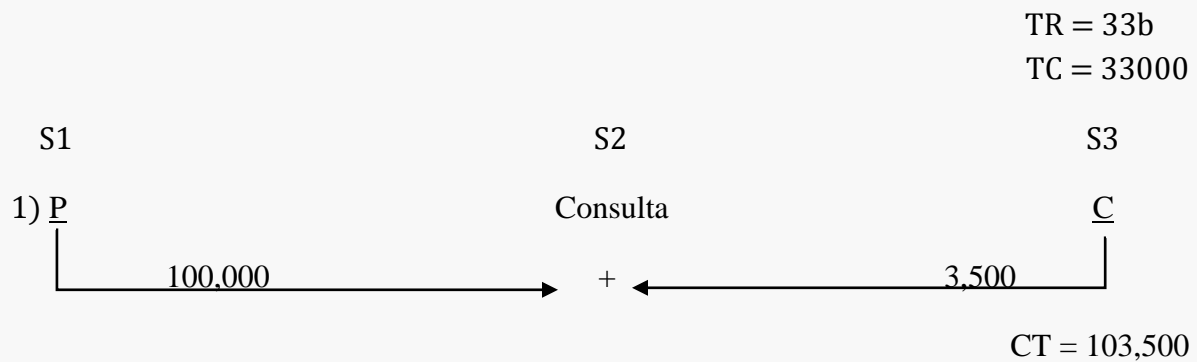
Consultas Distribuidas

1.-	S1	S2	S3
	<u>Profesor</u>	Consulta	<u>Curso</u>
Nombre = 15b	TR = 100b		TR = 50b
Matrícula = 10b	Reg = 1000		Reg = 70
Correo = 15b	100,000b		3,500b
Id_c = 8b			Id_curso = 8b
			Clave = 5b
			Ciclo = 20b

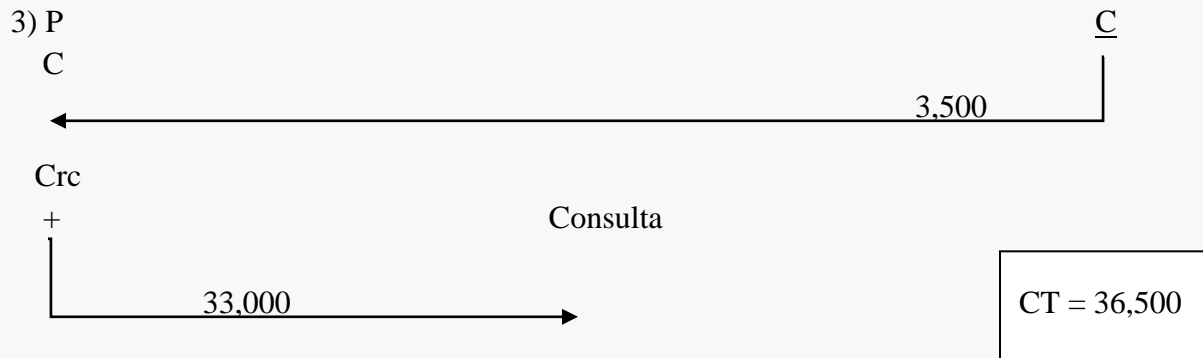
Consulta = Obtener por cada profesor su nombre, matrícula e id_curso a los cuales asistió.

AR = π nombre, matrícula, Id_c (Profesor \bowtie Id_c = Id_curso Curso)

SQL = Select nombre, matrícula, Id_curso
 From Profesor, Curso
 Where Id_c = Id_curso;



CT = 133,500

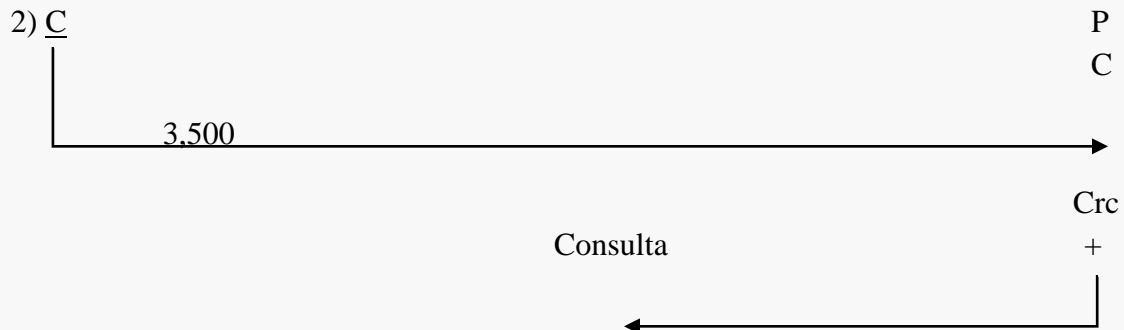
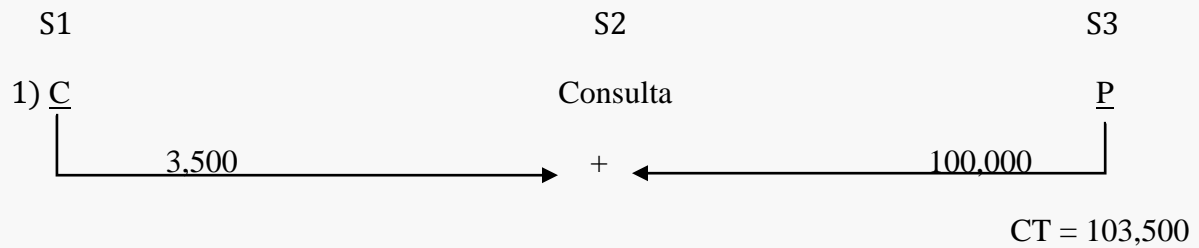


2.- Consulta = Obtener por cada curso su clave y matrícula del profesor.

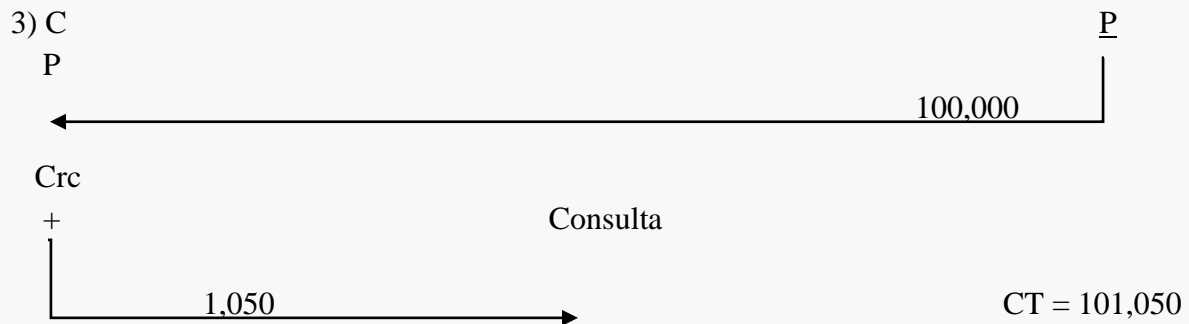
AR = π clave, matrícula (Curso \bowtie Id_curso = Id_c Profesor)

SQL = Select clave, matrícula
From Curso, Profesor
Where Id_curso = Id_c;

TR= 15b
TC = 1050



CT = 4,550



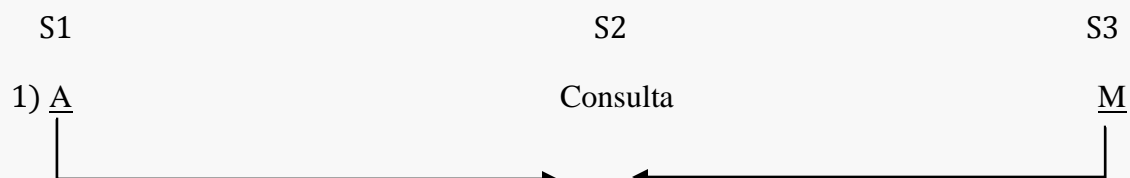
S3

Materia

TR = 800b	Nombre_m = 15b
Reg = 6000	Id_materia = 8b
480,000b	Especialidad = 12b

AR = π nombre, matrícula, Id_m (Alumno \bowtie Id_m = Id_materia Materia)

```
SQL = Select nombre, matrícula, Id_materia
      From Alumno, Materia
      Where Id_m = Id_materia;
```

$$\begin{aligned} \text{TR} &= 33b \\ \text{TC} &= 495,000 \end{aligned}$$


15'000,000

+

480,000

CT = 15'480,000

2) A

M

A

15'000,000

Consulta

Crc

+

495,000

CT 15'495,000

3) A

M

M

480,000

Crc

+

Consulta

495,000

CT = 975,000

Consultas Distribuidas Optimizadas

1.-

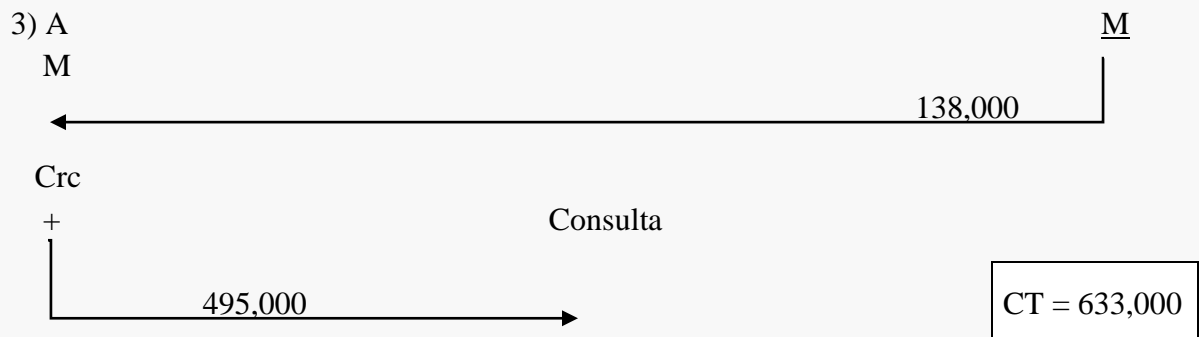
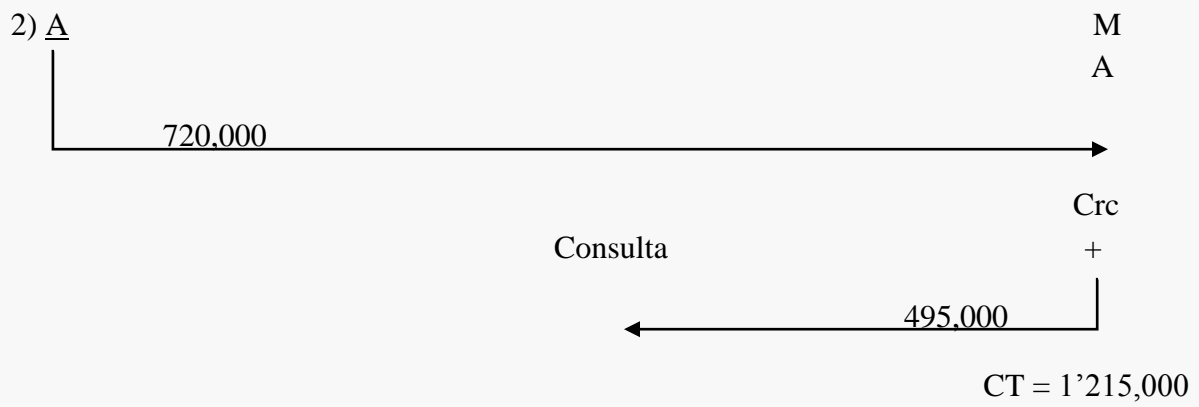
Select nombre, Matrícula, Id_m, tutor from Alumno;

π nombre, matrícula, Id_m, tutor (Alumno)

Select nombre_materia, Id_materia from Materia;

π nombre_materia, Id_materia (Materia)

Materia
138,000b

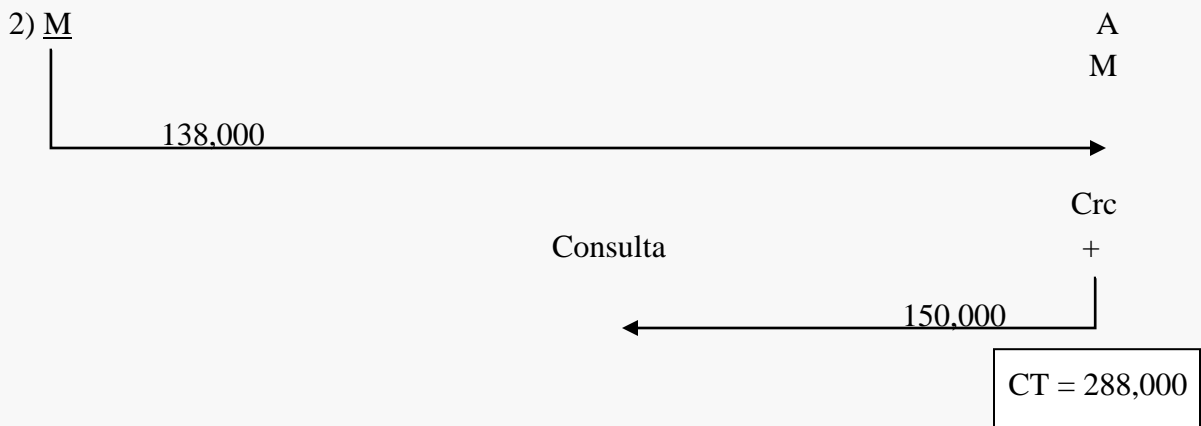


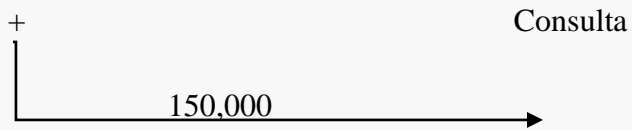
2.-

π nombre, matrícula, Id_m, tutor (Alumno)

 π nombre_materia, Id_materia (Materia)

Alumno
720,000b





CT = 870,000

Código

```
using SGBA.DataAccess.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Principal;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Security
{
    public class SGBAPrincipal : IPrincipal
    {
        IIdentity identity;

        public Usuario Usuario { get; set; }
        public IIdentity Identity
        {
            get
            {
                return identity;
            }
        }

        public SGBAPrincipal(Usuario usuario = null) {
            Usuario = usuario;
            identity = new SGBAIdentity(usuario);
        }
    }
}
```



```

        public bool IsInRole(string role)
        {
            var res = (Usuario != null && Usuario.Rol != null && Usuario.Rol.ToLower() ==
role.ToLower());

            return res;
        }

        public bool IsAdmin
        {
            get {
                var res = IsInRole("Admin");
                return res; }
        }
    }
}

```

```

using SGBA.DataAccess.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Security.Principal;
using System.Text;
using System.Threading.Tasks;

```

```

namespace SGBA.DataAccess.Security
{
    class SGBAIdentity : IIdentity
    {
        public SGBAIdentity(Usuario usuario = null) {
            Usuario = usuario;
        }
    }
}

```

```

public Usuario Usuario { get; set; }

public string AuthenticationType
{
    get
    {
        return "Custom";
    }
}

public bool IsAuthenticated
{
    get
    {
        return Usuario != null;
    }
}

public string Name
{
    get
    {
        return Usuario == null ? "" : Usuario.Nombre??"";
    }
}
}

```

```

using SGBA.DataAccess.Entities;
using System;
using System.Collections.Generic;
using System.Data.Entity;
using System.Data.Entity.Core.Objects;

```

```

using System.Data.SqlTypes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess
{
    public class SGBADataContext : DbContext
    {
        public SGBADataContext() : base("name=SGBAEntities")
        {
        }

        static SGBADataContext instance;

        public static SGBADataContext Instance
        {
            get
            {
                if (instance == null)
                {
                    instance = new SGBADataContext();
                }
                return instance;
            }
        }

        public DbSet<Profesor> Profesores { get; set; }
        public DbSet<Alumno> Alumnos { get; set; }
        public DbSet<Ciclo> Ciclos { get; set; }
        public DbSet<Curso> Cursos { get; set; }
        public DbSet<Especialidad> Especialidades { get; set; }
        public DbSet<Materia> Materias { get; set; }
        public DbSet<Persona> Personas { get; set; }
        public DbSet<Usuario> Usuarios { get; set; }
    }
}

```

```

public override int SaveChanges()
{
    foreach (var entry in this.ChangeTracker.Entries()) {
        if ((entry.State & (EntityState.Added | EntityState.Modified)) != 0) {
            foreach(var prop in entry.Entity.GetType().GetProperties()){
                if (prop.PropertyType == typeof(DateTime)) {
                    if ((DateTime)prop.GetValue(entry.Entity) <
SqlDateTime.MinValue.Value) {
                        prop.SetValue(entry.Entity, SqlDateTime.MinValue.Value);
                    }
                }
            }
        }
    }
    return base.SaveChanges();
}
}

```

```

using SGBA.DataAccess;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace ConsoleTest
{
    class Program
    {

```

```

static void Main(string[] args)
{
    SGBADataContext db = SGBADataContext.Instance;
    db.Ciclos.Add(new Ciclo()
    {
        Nombre = "Algo",
        FechaFin = DateTime.Now,
        FechaInicio = DateTime.Now.AddYears(-1)
    });
    db.SaveChanges();
    foreach (var item in db.Ciclos)
    {
        Console.WriteLine(item);
    }
    Console.ReadKey();
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Security.Cryptography;

```

```

namespace SGBA.DataAccess.Utils
{
    public static class CryptoUtils
    {

```

```

        public static string Encrypt(string inputString) {
            byte[] data = Encoding.ASCII.GetBytes(inputString);
            data = new SHA256Managed().ComputeHash(data);
            string hash = Encoding.ASCII.GetString(data);
            return hash;
        }
    }
}

using SGBA.DataAccess.Entities;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace SGBA.DataAccess.Security
{
    public static class Authenticator
    {
        public static bool AuthenticateUser(string username, string password)
        {
            try
            {
                if (username.ToLower() == "admin")
                {
                    var usr = new Usuario() { Nombre = "Admin", Rol = "Admin" };
                    System.Threading.Thread.CurrentPrincipal = new SGBAPrincipal(usr);
                    return true;
                }
            };
            var Db = SGBADataContext.Instance;
            var user = Db.Usuarios.Where(u => u.Nombre == username).FirstOrDefault();

```

```

        if (user == null) return false;
        if (user.Password == Utils.CryptoUtils.Encrypt(password))
        {

            System.Threading.Thread.CurrentPrincipal = new SGBAPrincipal(user);
            return true;
        }
        return false;
    }
    catch (Exception) {
        return false;
    }
}
}
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.SqlTypes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace SGBA.DataAccess.Entities
{
    [Table("Usuarios")]
    public class Usuario
    {

```

```

[Key]
public int Id { get; set; }
public string Nombre { get; set; }
public string Password { get; set; }
DateTime fechaCreacion;
public DateTime FechaCreacion
{
    get
    {
        return fechaCreacion < SqlDbType.MinValue.Value ?
SqlDateTime.MinValue.Value : fechaCreacion;
    }
    set { fechaCreacion = value; }
    public string Rol { get; set; }
}
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.SqlTypes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

```

```

namespace SGBA.DataAccess.Entities
{
    [Table("Personas")]
    public abstract class Persona

```



```

{
    [Key]
    public int Id { get; set; }
    public string Nombre { get; set; }
    public string ApellidoPaterno { get; set; }
    public string ApellidoMaterno { get; set; }
    DateTime fechaNacimiento;
    public DateTime FechaNacimiento {
        get
        {
            return fechaNacimiento < SqlDateTime.MinValue.Value ?
SqlDateTime.MinValue.Value : fechaNacimiento;
        }
        set { fechaNacimiento = value; }
    }
    public string Curp { get; set; }
    public string Email { get; set; }
    public string Telefono { get; set; }

    public string Matricula { get; set; }
    public int Edad {
        get
        {
            return (DateTime.Now - FechaNacimiento).Days / 365;
        }
    }
    public string NombreCompleto
    {
        get { return ToString(); }
    }
}

```

```

        public override string ToString()
        {
            return string.Format("{0} {1} {2} ({3})", Nombre, ApellidoPaterno,
ApellidoMaterno, Matricula);
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace SGBA.DataAccess.Entities
{
    public class Alumno: Persona
    {
        public Alumno() {
            Cursos = new List<Curso>();
        }
        public virtual Especialidad Especialidad { get; set; }
        public virtual Profesor Tutor { get; set; }
        public string Categoria { get; set; }
        public virtual List<Curso> Cursos { get; set; }
    }
}

```

```

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel.DataAnnotations.Schema;
using System.Data.SqlTypes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Entities
{
    public class Profesor : Persona
    {
        public string GradoAcademico { get; set; }
        DateTime fechaIngreso;
        public DateTime FechaIngreso
        {
            get
            {
                return fechaIngreso < SqlDateTime.MinValue.Value ?
                SqlDateTime.MinValue.Value : fechaIngreso;
            }
            set { fechaIngreso = value; }
        }
        public bool EsSuplente { get; set; }

        public virtual List<Materia> MateriasImpartidas { get; set; }

        public virtual List<Alumno> Tutorados { get; set; }

        public virtual List<Curso> Cursos { get; set; }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Entities
{
    [Table("Materias")]
    public class Materia
    {
        [Key]
        public int Id { get; set; }
        public string Clave { get; set; }
        public string Nombre { get; set; }
        public virtual List<Especialidad> Especialidades { get; set; }
        public virtual List<Curso> Cursos { get; set; }
        public virtual List<Profesor> Profesores { get; set; }
        public override string ToString()
        {
            return string.Format("{0}-{1}", Clave, Nombre);
        }
    }
}

using System;
using System.Collections.Generic;

```

```

using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Entities
{
    [Table("Especialidades")]
    public class Especialidad
    {
        public Especialidad() {
            Materias = new List<Materia>();
        }
        [Key]
        public int Id { get; set; }
        public string Nombre { get; set; }
        public string Clave { get; set; }
        public virtual List<Materia> Materias { get; set; }
        public override string ToString()
        {
            return string.Format("{0}-{1}",Clave,Nombre);
        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Linq;

```

```

using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Entities
{
    [Table("Cursos")]
    public class Curso
    {
        public Curso()
        {
            Alumnos = new List<Alumno>();
        }
        [Key]
        public int Id { get; set; }
        public string Clave { get; set; }
        public virtual Ciclo CicloEscolar { get; set; }
        public virtual Materia Materia { get; set; }
        public virtual Profesor Profesor { get; set; }
        public virtual List<Alumno> Alumnos { get; set; }
        public string Especialidades
        {
            get
            {
                return Materia.Especialidades
                    .Select(e => e.Nombre)
                    .Aggregate((s1, s2) => { return s1 + ", " + s2; });
            }
        }
    }
}

```

```

using System;
using System.Collections.Generic;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
using System.Data.SqlTypes;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace SGBA.DataAccess.Entities
{
    [Table("Ciclos")]
    public class Ciclo
    {
        [Key]
        public int Id { get; set; }
        public string Nombre { get; set; }
        public int Año { get; set; }
        DateTime fechaInicio;
        public DateTime FechaInicio
        {
            get
            {
                return fechaInicio < SqlDateTime.MinValue.Value ?
SqlDateTime.MinValue.Value : fechaInicio;
            }
            set { fechaInicio = value; }
        }
        DateTime fechaFin;
    }
}

```

```

public DateTime FechaFin {
    get
    {
        return fechaFin < SqlDbType.MinValue.Value ? SqlDbType.MinValue.Value
: fechaFin;
    }
    set { fechaFin = value; }
}

public virtual List<Curso> Cursos { get; set; }

public Ciclo()
{
    Cursos = new List<Curso>();
    FechaInicio = FechaFin = DateTime.Now;
}

public override string ToString()
{
    return Id + " - " + Nombre;
}
}
}

```

Pantallasos

 Login SGBA ✕




Nombre de Usuario:

Contraseña:

Cancelar

Ingresar

 Login SGBA ✕



Nombre de Usuario:

admin

Contraseña:

•••••

Cancelar

Ingresar


Login SGBA




Nombre de Usuario:

Contraseña:




Usuario o contraseña incorrectos



Establecer contraseña


Contraseña:


Confirmación:

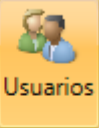

SGBA


Principal



Agregar



Modificar



Eliminar



Usuarios



Alumnos


Profesores


Especialidad


Materias


Cursos


Ciclos

Acciones
Catalogo

Usuarios

Id	Nombre	Rol	Creación
1	Juan	Admin	5/8/2016 12:36:25
2	Pedro	Director	5/8/2016 12:51:25

SGBA

Principal

Agregar Modificar Eliminar

Usuarios Alumnos Profesores Especialidad Materias Cursos Ciclos

Acciones Catalogo

Usuarios

Id	Nombre	Rol	Creación
1	Juan	Admin	5/8/2016 12:36:25
2	Pedro	Director	5/9/2016 12:21:52

SGBA

Guardar

Actualizar

Salir

Materias Cursos Ciclos

1	Juan	Admin	5/8/2016 12:36:25
2	Pedro	Director	5/9/2016 12:21:52

Catálogo de Alumno

Alumnos

Nombre:

Apellido Paterno:

Apellido Materno:

Fecha Nacimiento: 19

Curp:

Email:

Telefono:

Matricula:

Categoria:

Tutor:

Especialidad:

Alumnos

Nombre:

Apellido Paterno:

Apellido Materno:

Fecha Nacimiento: 19

Curp:

Email:

Telefono:

Matricula:

Categoria:

Tutor:

Especialidad:

Catálogo de Profesor

Profesor

Nombre:

Apellido Paterno:

Apellido Materno:

Fecha Nacimiento: 19

Curp:

Email:

Telefono:

Matricula:

Grado académico:

Fecha de Ingreso: 19

Suplente: ☐

Profesor

Nombre: Roberto

Apellido Paterno: Gutierrez

Apellido Materno: Alvarez

Fecha Nacimiento: 12/02/1980

Curp: GUAR800212HJCRSD12

Email: robert.sanc@gmail.com

Telefono: 3313412580

Matricula: 1234

Grado académico: PHD en artes

Fecha de Ingreso: 13/06/1995

Suplente: ☐

Cancelar Aceptar

Catálogo de Ciclo

Insertar C...

Noml:

Año: 0

Fecha: 08/05/2016

Fecha: 08/05/2016

Cancelar Aceptar

Insertar C...

Noml: cicloA

Año: 2016

Fecha: 18/01/2016

Fecha: 27/05/2016

Cancelar Aceptar

Triggers

```
1.- CREATE TRIGGER insertaAlumno
  AFTER INSERT
  ON alumno
  FOR EACH ROW
  INSERT INTO
```

(matricula,tutor,especialidad,categoria,fecha_nac,edad,telefono,correo,nombre,curp)

```
VALUES(NEW.matricula,NEW.tutor,NEW.especialidad,NEW.categoria,NEW.fecha_nac,  
NEW.edad,NEW.telefono,NEW.correo,NEW.nombre,NEW.curp);
```

2.- CREATE TRIGGER modificarProfesor

BEFORE UPDATE

ON maestro

FOR EACH ROW

```
UPDATE(matricula,sueldo,antiguedad,materia,fecha_nac,edad,telefono,correo,nombre,curp  
)VALUES(OLD.matricula,OLD.sueldo,OLD.antiguedad,OLD.materia,OLD.fecha_nac,OLD.  
D.edad,OLD.telefono,OLD.correo,OLD.correo,OLD.curp);  
WHERE matricula=OLD.matricula;
```

3.- CREATE TRIGGER borrarAlumno

BEFORE DELETE

ON alumno

FOR EACH ROW

```
(matricula,tutor,especialidad,categoria,fecha_nac,edad,telefono,correo,nombre,curp)  
VALUES (OLD.matricula, OLD.tutor, OLD.especialidad, OLD.categoria, OLD.fecha_nac,  
OLD.edad, OLD.telefono, OLD.correo, OLD.nombre, OLD.curp);  
WHERE matricula=OLD.matricula;
```

Conclusiones

Díaz Márquez Oscar

El desarrollar un sistema requiere de mucho tiempo y dedicación, ya que no solo es realizar el código que se ejecutará, si no que antes de eso, está la documentación adecuada y la elaboración de manuales (técnico y de usuario) que van de la mano con el software terminado.

Cada aspecto de la documentación es importante de realizar para que el software tenga un correcto funcionamiento de acuerdo a los requisitos pedidos por el cliente.

Este proyecto me sirvió para darme una idea de cómo será el proceso de entregar un proyecto ya como un profesional, y de tener que poner atención a cada uno de los detalles que implica un proyecto, desde los requerimientos, hasta los tiempos que se deben de manejar al elaborar cada punto o etapa del proyecto.

Garfia Pahua José Guadalupe

El desarrollo de este sistema me permitió darme una idea como sería más o menos trabajar desarrollando software, con tiempos de entrega, avances y sobre todo, me gusto el proceso que se debe de seguir para desarrollar trabajos de este tipo, además tuve la oportunidad de conocer también un poco más a mi compañero de equipo.

En lo personal, es el primer trabajo que realizó de manera semi-profesional por decirlo de alguna manera, y me siento bien porque no solo me agrado mucho hacer este proyecto, sino que también pude darme cuenta de lo difícil que es desarrollar software de buena calidad, el proyecto SGBA es la combinación de dos de las materias que más me han gustado en lo que va de esta licenciatura, es decir, manejamos programación estructurada y bases de datos, que le dan a nuestro sistema un toque profesional como también un enfoque particular.