Técnicas Avanzadas de Programación

Profesora: Antonieta Kuz

Alumno: Nuñez Américo Emilio

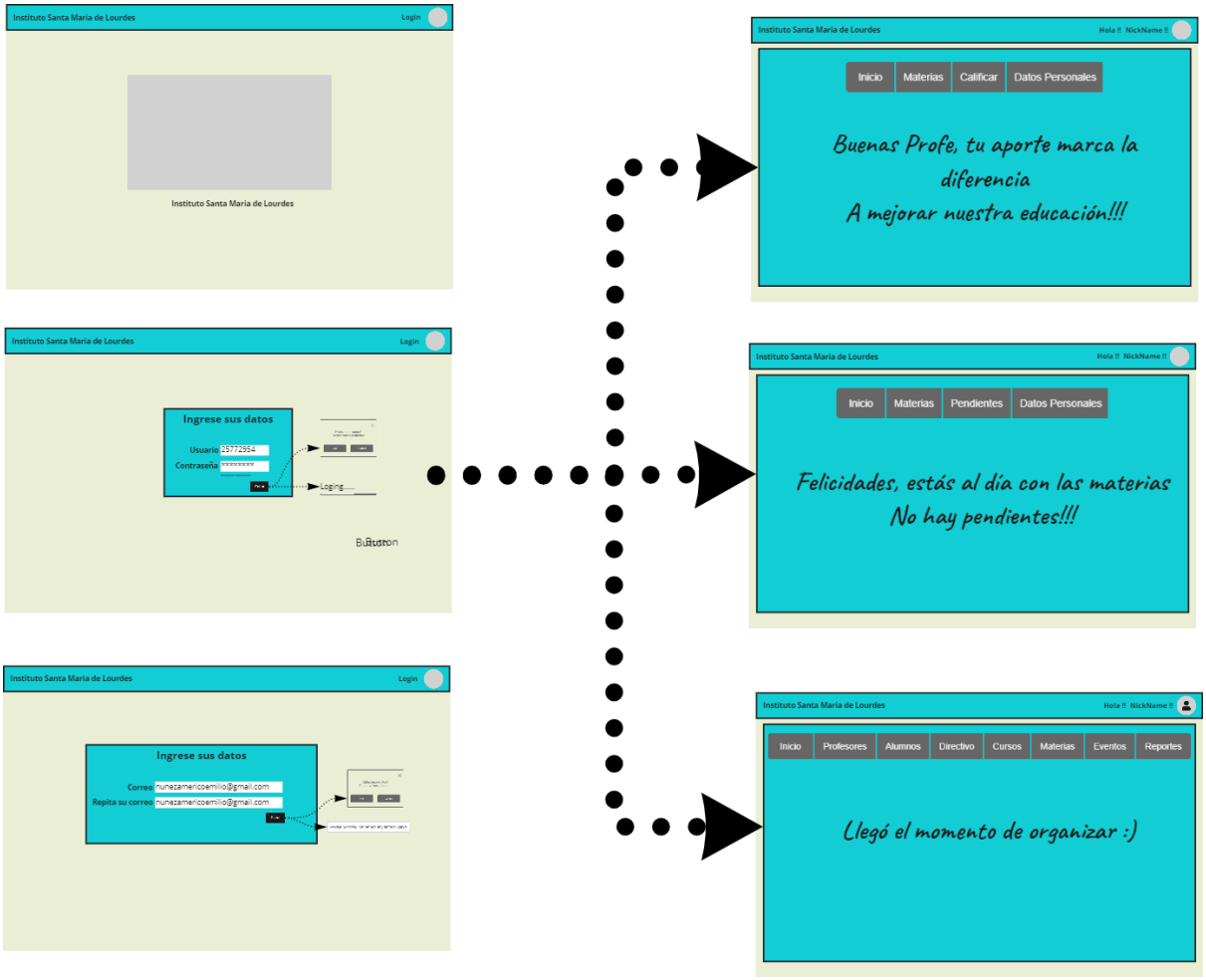
Año: 2022

Cursada: 1er cuatrimestre

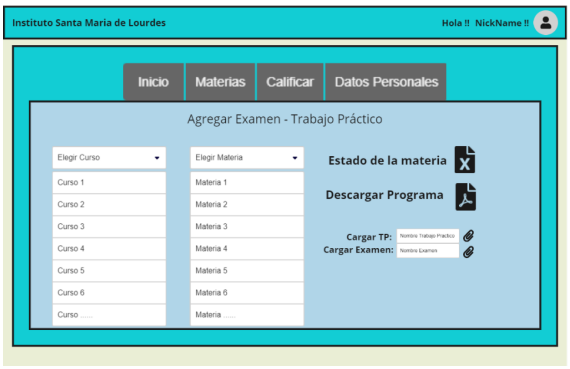
ÍNDICE

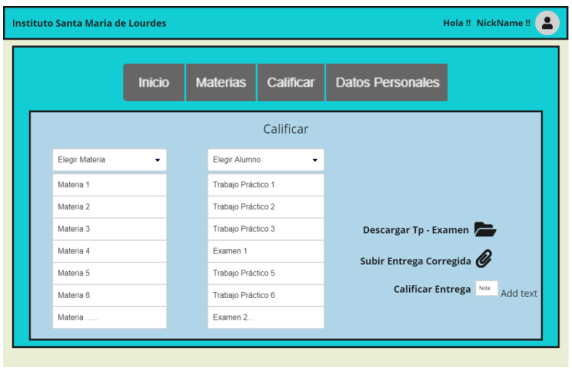
1. Maquetado del proyecto

Login de los tres Roles

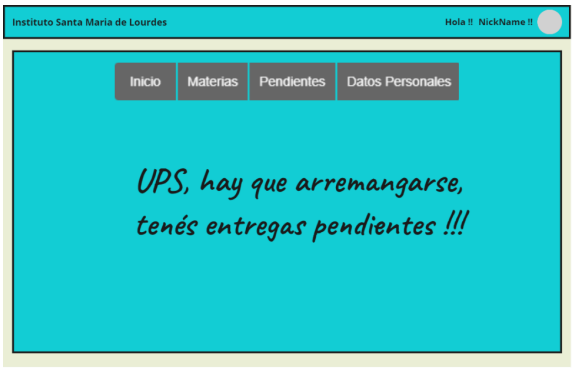


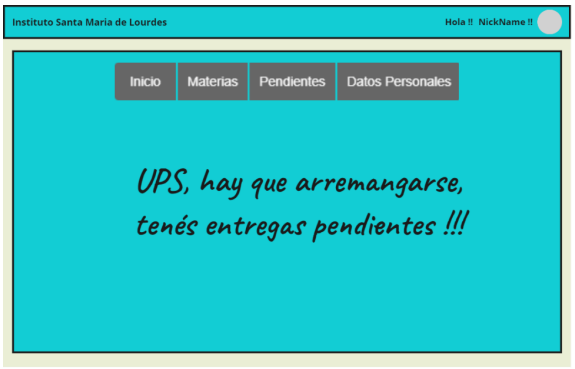
Funcionalidades Rol Profesor



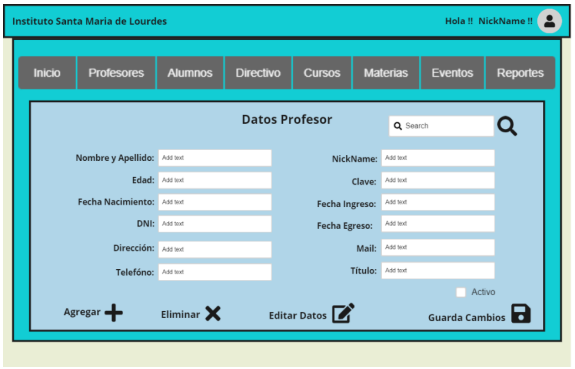


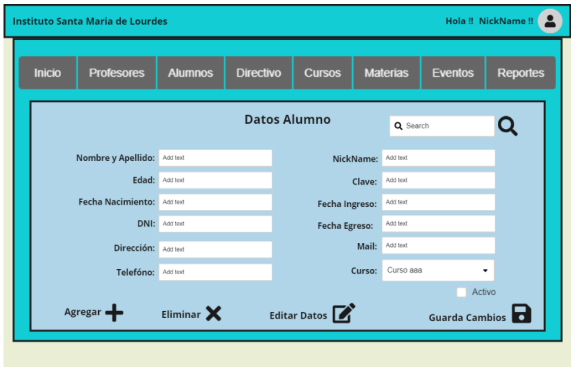
Funcionalidades Rol Alumno

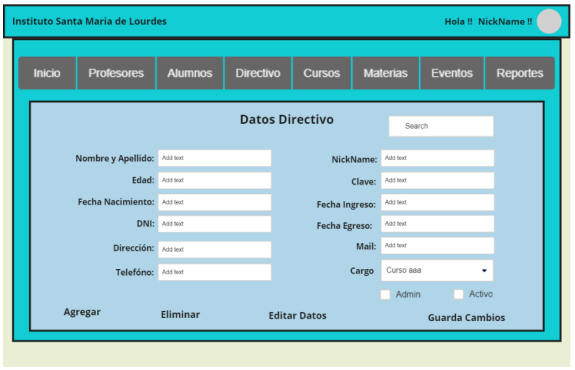




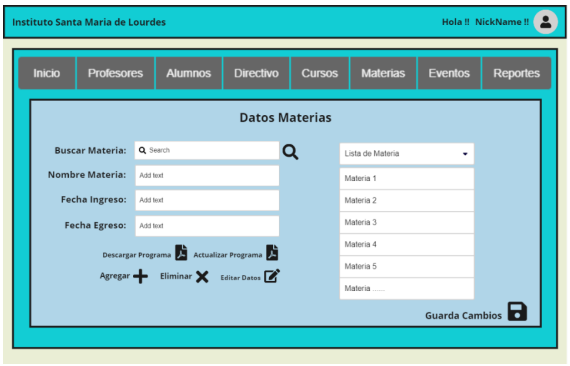
Funcionalidades Rol Directivo

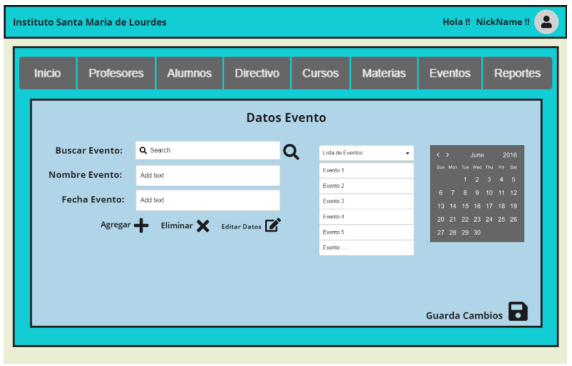





Instituto Santa Maria de Lourdes
Hola !! NickName !!
Inicio
Profesores
Alumnos
Directivo
Cursos
Materias
Eventos
Reportes
Datos Directivo
Search
Nombre y Apellido:
Edad:
Fecha Nacimiento:
DNI:
Dirección:
Telefóno:
NickName:
Clave:
Fecha Ingreso:
Fecha Egreso:
Mail:
Cargo
Curso aaa
Admin
Activo
Agregar
Eliminar
Editar Datos
Guarda Cambios

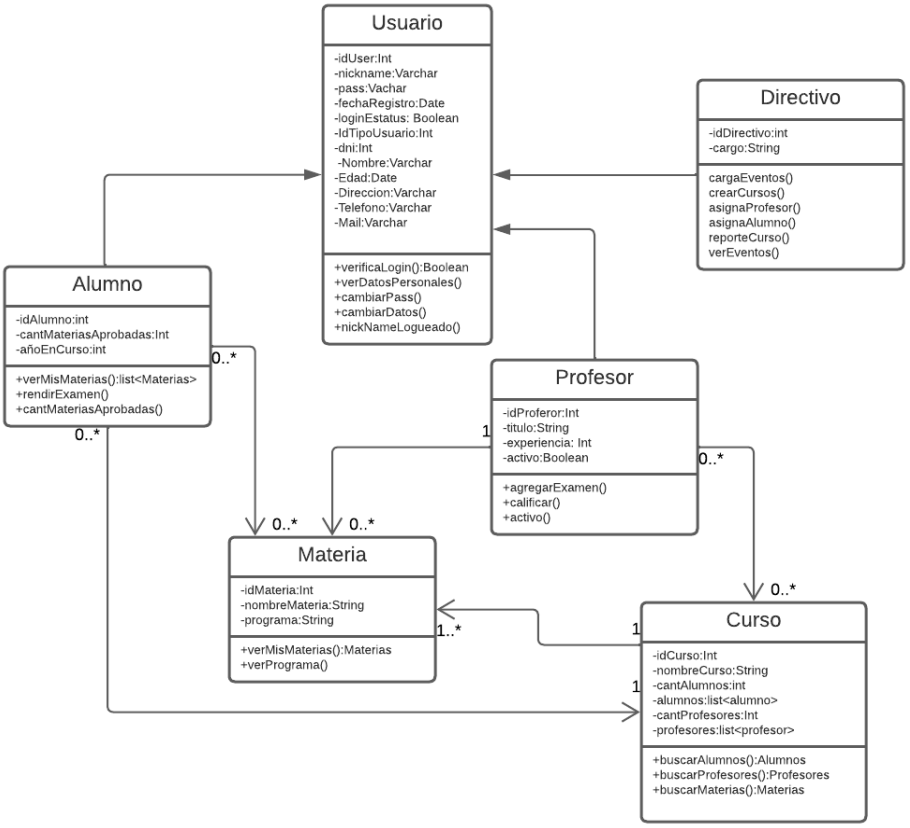

Instituto Santa Maria de Lourdes
Hola !! NickName !!
Inicio
Profesores
Alumnos
Directivo
Cursos
Materias
Eventos
Reportes
Datos Curso
Nombre del Curso:
Año:
Fecha Creación:
Fecha Baja:
Lista Profesores:
Nombre Profesor
Lista Alumnos:
Nombre Alumno
Lista Materias:
Nombre Materias
Buscar Curso:
Buscar Profesor:
Buscar Alumno:
Buscar Materia:
Agregar
Eliminar
Editar Datos
Guarda Cambios


Instituto Santa Maria de Lourdes
Hola !! NickName !!
Inicio
Profesores
Alumnos
Directivo
Cursos
Materias
Eventos
Reportes
Datos Materias
Buscar Materia:
Nombre Materia:
Fecha Ingreso:
Fecha Egreso:
Lista de Materia
Materia 1
Materia 2
Materia 3
Materia 4
Materia 5
Materia .....
Descargar Programa
Actualizar Programa
Agregar
Eliminar
Editar Datos
Guarda Cambios

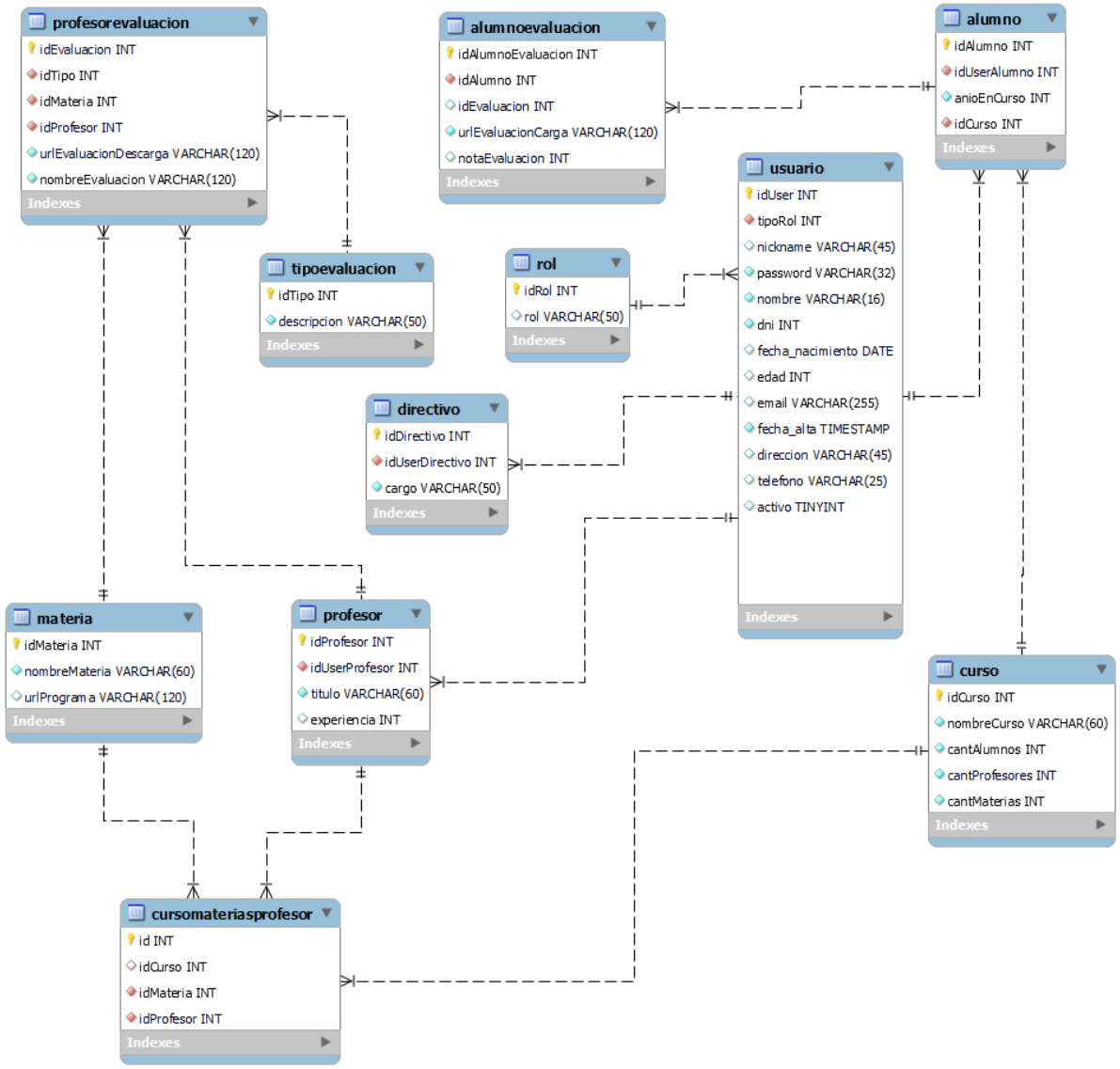

Instituto Santa Maria de Lourdes
Hola !! NickName !!
Inicio
Profesores
Alumnos
Directivo
Cursos
Materias
Eventos
Reportes
Datos Evento
Buscar Evento:
Nombre Evento:
Fecha Evento:
Lista de Eventos
Evento 1
Evento 2
Evento 3
Evento 4
Evento 5
Evento .....
Agregar
Eliminar
Editar Datos
Guarda Cambios


Instituto Santa Maria de Lourdes
Hola !! NickName !!
Inicio
Profesores
Alumnos
Directivo
Cursos
Materias
Eventos
Reportes
Generar Reporte
Fecha Desde:
Fecha Hasta:
Repositorio destino:
Generar Reporte

2. Diagrama de Clases



3. Diagrama de Entidad Relación



4. Tablas de la base de datos con sus ids y atributos

```
CREATE SCHEMA `instituto_smdl`;
use instituto_smdl

# Tabla tipo de roles
CREATE TABLE rol (idRol int not null,
                  rol varchar(50),
                 primary key (idRol)
                )

# Tabla tipo de evaluciones
CREATE TABLE tipoEvaluacion(idTipo int not null,
                            descripcion varchar(50) not null,
                           primary key(idTipo)
                          )
# Tabla usuarios
# DROP TABLE USUARIO
CREATE TABLE USUARIO(dni INT NOT NULL,
                     idRol INT NOT NULL,
                     nickname VARCHAR(50) NOT NULL,
                     password VARCHAR(50) NOT NULL,
                     nombre VARCHAR(120) NOT NULL,
                     fechaNacimiento DATE NOT NULL,
                     edad INT NOT NULL,
                     email VARCHAR(120),
                     fechaAlta TIMESTAMP,
                     direccion VARCHAR(120),
                     telefono VARCHAR(50),
                     activo BOOLEAN,
                     primary key(dni),
                     foreign key (idRol) references ROL(idRol)
                    )

# DROP TABLE CURSO
CREATE TABLE CURSO (idCurso INT NOT NULL,
                    nombreCurso VARCHAR(120) NOT NULL,
                    cantAlumnos INT NOT NULL,
                    cantProfesores INT NOT NULL,
                    primary key (idCurso)
                   )

# tabla alumnos
# DROP TABLE ALUMNO
CREATE TABLE ALUMNO(dniAlumno INT NOT NULL,
                    anioEnCurso INT NOT NULL,
                    idCurso INT NOT NULL,
                    primary key(dniAlumno),
                    foreign key (dniAlumno) references USUARIO(dni),
                    foreign key (idCurso) references CURSO(idCurso)
                    )
```

```
# tabla profesores
# DROP TABLE PROFESOR
CREATE TABLE PROFESOR(dniProfesor int not null,
                      titulo varchar(120) not null,
                      experiencia int null,
                      primary key (dniProfesor),
                      foreign key (dniProfesor) references USUARIO(dni)
                     )

# TABLA DIRECTIVOS
# DROP TABLE DIRECTIVO
CREATE TABLE DIRECTIVO(dniDirectivo INT NOT NULL,
                       cargo VARCHAR(120) NOT NULL,
                       primary key (dniDirectivo),
                       foreign key (dniDirectivo) references USUARIO(dni)
                       )
#DROP TABLE MATERIA
CREATE TABLE MATERIA(idMateria int not null,
                     nombreMateria varchar (120) not null,
                     urlPrograma varchar(250) null,
                     primary key (idMateria)
                    )

# Tabla que contiene evaluacion de los docentes
# DROP TABLE profesorEvaluacion
CREATE TABLE profesorEvaluacion(idEvaluacion int not null,
                                idTipo int not null,
                                idCurso int not null,
                                idMateria int not null,
                                dniProfesor int not null,
                                urlEvaluacionCarga varchar(250) not null,
                                urlEvaluacionDescarga varchar(250) not null,
                                nombreEvaluacion varchar(120) not null,
                                primary key(idEvaluacion),
                                foreign key (idTipo) references tipoEvaluacion(idTipo),
                                foreign key (idMateria) references MATERIA(idMateria),
                                foreign key (idCurso) references CURSO(idCurso),
                                foreign key (dniProfesor) references,
                                PROFESOR(dniProfesor)

                                )
#DROP TABLE CursoMateriaProfesor
CREATE TABLE cursoMateriaProfesor (id  int not null,
                                   idCurso int null,
                                   idMateria int not NULL,
                                   dniProfesor int not null,
                                   primary key (id),
                                   foreign key (idMateria) references MATERIA(idMateria),
                                   foreign key (idCurso) references CURSO(idCurso),
                                   foreign key (dniProfesor) references
                                   PROFESOR(dniProfesor)
                                   )
```

```
#drop table alumnoevaluacion
CREATE TABLE alumnoEvaluacion (idAlumnoEvaluacion  int not null,
                                dniAlumno int not null,
                                idEvaluacion int not null,
                                notaEvaluacion int NULL,
                                primary key (idAlumnoEvaluacion),
                               foreign key (dniAlumno) references alumno(dniAlumno),
                               foreign key (idEvaluacion) references
                               profesorEvaluacion(idEvaluacion)
                               )
```

5. Stores procedures

```
USE Instituto_smdl
/************************************************************/
/************** SP Tipos de ROLES **********************/
/************************************************************/
DELIMITER //
# SP para insertar los roles
DROP PROCEDURE IF EXISTS spInsertRol;
CREATE PROCEDURE `spInsertRol` (IN idRol INT, IN rol VARCHAR (50))
BEGIN
        INSERT INTO rol(IdRol,rol) values (IdRol,rol) ;
END //
DELIMITER ;

DELIMITER //
# SP para que me devuelva todos los roles
DROP PROCEDURE IF EXISTS spGetRoles;
CREATE PROCEDURE `spGetRoles` ()
BEGIN
        SELECT * FROM rol;
END //
DELIMITER ;

DELIMITER //
# SP para eliminar un rol
DROP PROCEDURE IF EXISTS spDeleteRol;
CREATE PROCEDURE `spDeleteRol` (IN idRol INT)
BEGIN
        DELETE FROM rol R WHERE r.idRol = idRol;
END //
DELIMITER ;

DELIMITER //
# SP para actualizar descripcion de un rol
DROP PROCEDURE IF EXISTS spUpdateRol;
CREATE PROCEDURE `spUpdateRol` (IN idRol INT,IN rol VARCHAR(50))
BEGIN
        UPDATE rol r SET r.rol = rol  WHERE r.idRol = idRol;
END //
```

```
DELIMITER ;

/*************************************************************/
/************** SP Tipo de Evaluaciones **********************/
/*************************************************************/
DELIMITER //
# SP para insertar los tipos de evaluaciones
DROP PROCEDURE IF EXISTS spInsertTipoEvaluacion;
CREATE PROCEDURE `spInsertTipoEvaluacion` (IN idTipo INT, IN descripcion VARCHAR (50))
BEGIN
	INSERT INTO tipoevaluacion(IdTipo, descripcion) values (IdTipo, descripcion) ;
END //
DELIMITER  ;

# SP para que me devuelva los datos de un tipo de evaluacion
DELIMITER //
#DROP PROCEDURE IF EXISTS spGetTipoEvaluacion;
CREATE PROCEDURE spGetTipoEvaluacion(IN tipo INT)
BEGIN
	SELECT * FROM tipoevaluacion WHERE idTipo = tipo;
END //
DELIMITER  ;

DELIMITER //
# SP para que me devuelva todos los tipos de evaluaciones
DROP PROCEDURE IF EXISTS spGetAllTipoEvaluacion;
CREATE PROCEDURE spGetAllTipoEvaluacion()
BEGIN
	SELECT * FROM tipoevaluacion ;
END //
DELIMITER  ;

DELIMITER //
# SP para eliminar un tipo de evaluacion
DROP PROCEDURE IF EXISTS spDeleteTipoEvaluacion;
CREATE PROCEDURE `spDeleteTipoEvaluacion` (IN idTipo INT)
BEGIN
	DELETE FROM tipoEvaluacion e WHERE e.idTipo = idTipo;
END //
DELIMITER ;

DELIMITER //
# SP para actualizar descripcion de un rol
DROP PROCEDURE IF EXISTS spUpdateTipoEvaluacion;
CREATE PROCEDURE `spUpdateTipoEvaluacion` (IN idTipo INT, IN descripcion VARCHAR(50))
BEGIN
	UPDATE tipoEvaluacion e SET e.descripcion = descripcion  WHERE e.idTipo = idTipo;
END //
DELIMITER ;
```

```
/************************************************************/
/**************        SP MATERIA      **********************/
/************************************************************/
DELIMITER //
# SP para insertar MATERIAS
DROP PROCEDURE IF EXISTS spInsertMateria;
CREATE PROCEDURE `spInsertMateria` (IN idMateria INT, IN nombreMateria VARCHAR(120), IN
urlPrograma VARCHAR(250))
BEGIN
        INSERT INTO MATERIA(idMateria, nombreMateria, urlPrograma)
        VALUES (idMateria, nombreMateria, urlPrograma);
END //
DELIMITER ;

# SP para ver los datos de una materia
DELIMITER //
DROP PROCEDURE IF EXISTS spGetMateria ;
CREATE PROCEDURE `spGetMateria`(IN idMateria INT)
BEGIN
        SELECT * FROM MATERIA m WHERE m.idMateria = idMateria;
END //
DELIMITER ;

# SP para ver todas las materias
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAllMateria ;
CREATE PROCEDURE `spGetAllMateria`()
BEGIN
        SELECT * FROM MATERIA;
END //
DELIMITER ;

# SP para eliminar una materia
DELIMITER //
DROP PROCEDURE spDeleteMateria;
CREATE PROCEDURE spDeleteMateria(IN idMateria INT)
BEGIN
        DELETE FROM MATERIA m WHERE m.IdMateria = idMateria;
END //
DELIMITER ;

# SP para actualizar Nombre materia
# UPDATE artículos SET iva = ’18’, dto = ‘5’ WHERE familia = «colchones»
# UPDATE tabla_x SET registro_x = IF(A IS NOT NULL, A, IF(B IS NOT NULL, B, C));
```

```
DELIMITER //
#DROP PROCEDURE `instituto_smdl`.`spUpdateMateria`;

CREATE PROCEDURE spUdateMateria(IN id INT, IN newNombre VARCHAR(120), IN newUrl
VARCHAR(250))
BEGIN
	UPDATE MATERIA
	SET nombreMateria = IF(newNombre IS NOT NULL, newNombre, nombreMateria),
	urlPrograma = IF(newUrl IS NOT NULL, newUrl, urlPrograma)
	 WHERE idMateria = id;
END //
DELIMITER ;

/************************************************************/
/*************        SP CURSO      * *********************/
/************************************************************/
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spInsertRol`;
CREATE PROCEDURE spInsertCurso(IN IdCurso INT, IN NombreCurso VARCHAR(120), IN
cantAlumnos INT, IN cantProfesores INT)
BEGIN
	INSERT INTO CURSO (IdCurso, NombreCursO, cantAlumnos, cantProfesores)
	VALUES (IdCurso, NombreCursO, cantAlumnos, cantProfesores);
END //
DELIMITER ;

# SP para eliminar un curso
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spDeleteCurso`;
CREATE PROCEDURE spDeleteCurso(IN id INT)
BEGIN
	DELETE FROM CURSO WHERE IdCurso =  id;
END //
DELIMITER ;

# SP PARA ACTUALIZAR DATOS DE UN CURSO
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spUpdateCurso`;
CREATE PROCEDURE spUpdateCurso(IN id INT, IN newNombre VARCHAR(120), IN
newCantAlumnos INT, IN newCantProfesores INT)
BEGIN
	UPDATE CURSO
	SET nombreCurso = IF(newNombre IS NOT NULL, newNombre, nombreCurso),
		cantAlumnos = IF(newCantAlumnos IS NOT NULL, newCantAlumnos, cantAlumnos),
	cantProfesores = IF(newCantProfesores IS NOT NULL, newCantProfesores, cantProfesores)
	WHERE idCurso = id;
END //
DELIMITER ;
```

```
# SP PARA DEVOLVER UN CURSO
DELIMITER //
#DROP PROCEDURE `instituto_smdl`.`spGetCurso`;
CREATE PROCEDURE spGetCurso(IN id INT)
BEGIN
	SELECT * FROM CURSO  WHERE idCurso = id;
END //
DELIMITER ;

# SP PARA DEVOLVER TODOS LOS DATOS DE LOS CURSOS
DELIMITER //
#DROP PROCEDURE `instituto_smdl`.`spGetAllCurso`;
CREATE PROCEDURE spGetAllCurso(IN id INT)
BEGIN
	SELECT * FROM CURSO;
END //
DELIMITER ;

/************************************************************/
/**************        SP PROFESOR     **********************/
/************************************************************/

# SP PARA INSERTAR UN PROFESOR
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spInsertProfesor`;
CREATE PROCEDURE spInsertProfesor(IN dniProf INT, IN titu VARCHAR(120), IN exper INT)
BEGIN
	INSERT INTO PROFESOR (dniProfesor, titulo, experiencia)
	VALUES(dniProf, titu, exper);
END //
DELIMITER ;

# SP PARA DEVOLVER DATOS DE UN PROFESOR
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spGetProfesor`;
CREATE PROCEDURE spGetProfesor(IN dniProf INT)
BEGIN
	SELECT * FROM PROFESOR WHERE dniProfesor = dniprof;
END //
DELIMITER ;

# SP PARA DEVOLVER TODOS LOS PROFESORES
DELIMITER //
#DROP PROCEDURE `instituto_smdl`.`spGetAllProfesor`;
CREATE PROCEDURE spGetAllProfesor()
BEGIN
	SELECT * FROM PROFESOR;
END //
DELIMITER ;
```

```
# SP PARA ACTUALIZAR DATOS DE UN PROFESOR
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spUpdateProfeTituExper`;
CREATE PROCEDURE spUpdateProfeTituExper(IN oldDniProf INT , IN newdniProf INT, IN titu
VARCHAR(120), IN exper INT)
BEGIN
	UPDATE PROFESOR
	SET dniProfesor = IF(newDniProf IS NOT NULL, newDniProf, dniProfesor),
		titulo = IF(titu IS NOT NULL, titu, titulo),
		experiencia = IF(exper IS NOT NULL, exper, experiencia)
	WHERE dniProfesor = oldDniProf;
END //
DELIMITER ;

/************************************************************/
/**************        SP ALUMNO       **********************/
/************************************************************/
# SP PARA INSERTAR UN ALUMNO
DELIMITER //
DROP PROCEDURE IF EXISTS spInsertAlumno;
CREATE PROCEDURE spInsertAlumno(IN dniAlum INT, IN anio INT, IN newIdCurso INT)
BEGIN
	INSERT INTO ALUMNO (dniAlumno, anioEnCurso, idCurso)
	VALUES(dniAlum, anio, newIdCurso);
END //
DELIMITER ;

# SP PARA DEVOLVER DATOS DE UN ALUMNO
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAlumno;
CREATE PROCEDURE spGetAlumno(IN dniAlum INT)
BEGIN
	SELECT * FROM ALUMNO WHERE dniAlumno = dniAlum;
END //
DELIMITER ;

# SP PARA DEVOLVER TODOS LOS ALUMNOS
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAllAlumno;
CREATE PROCEDURE spGetAllAlumno()
BEGIN
	SELECT * FROM ALUMNO;
END //
DELIMITER ;


# SP PARA ACTUALIZAR DATOS DE UN ALUMNO
DELIMITER //
DROP PROCEDURE IF EXISTS spUpdateAlumAnioCurso;
CREATE PROCEDURE spUpdateAlumAnioCurso(IN oldDniAlum INT , IN newDniAlum INT, IN
newAnio INT, IN newCurso INT)
BEGIN
```

```
        UPDATE ALUMNO
        SET dniAlumno = IF(newDniAlum IS NOT NULL, newDniAlum, dniAlumno),
            anioEnCurso = IF(newDniAlum IS NOT NULL, newDniAlum, anioEnCurso),
            idCurso = IF(newCurso IS NOT NULL, newCurso, idCurso)
        WHERE dniAlumno = oldDniAlum;
END //
DELIMITER ;

/************************************************************/
/**************        SP DIRECTIVO    **********************/
/************************************************************/
# SP PARA INSERTAR UN DIRECTIVO
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spInsertDirectivo`;
CREATE PROCEDURE spInsertDirectivo(IN dniDire INT, IN cargo VARCHAR(120))
BEGIN
        INSERT INTO DIRECTIVO (dniDirectivo, cargo)
        VALUES(dniDire, cargo);
END //
DELIMITER ;

# SP PARA DEVOLVER DATOS DE UN DIRECTIVO
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spGetDirectivo`;
CREATE PROCEDURE spGetDirectivo(IN dniDire INT)
BEGIN
        SELECT * FROM DIRECTIVO WHERE dniDirectivo = dniDire;
END //
DELIMITER ;

# SP PARA DEVOLVER TODOS LOS DIRECTIVO
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spGetAllDirectivo`;
CREATE PROCEDURE spGetAllDirectivo()
BEGIN
        SELECT * FROM DIRECTIVO;
END //
DELIMITER ;

# SP PARA ACTUALIZAR DATOS DE UN DIRECTIVO
DELIMITER //
DROP PROCEDURE `instituto_smdl`.`spUpdateProfeTituExper`;
CREATE PROCEDURE spUpdateDireCargo(IN oldDniDire INT , IN newDniDire INT, IN newCargo
VARCHAR(120))
BEGIN
        UPDATE DIRECTIVO
        SET dniDirectivo = IF(newDniDire IS NOT NULL, newDniDire, dniDirectivo),
            cargo = IF(newCargo IS NOT NULL, newCargo, cargo)
        WHERE dniDirectivo = oldDniDire;
END //
DELIMITER ;
```

```
/************************************************************/
/**************      SP USUARIO      **********************/
/************************************************************/

# SP para insertar un usuario
DELIMITER //
DROP PROCEDURE IF EXISTS spInsertUsuario;
CREATE PROCEDURE spInsertUsuario(IN newDni INT, IN newIdRol INT, IN newNickname
VARCHAR(50), IN newClave VARCHAR(50), IN newNombre VARCHAR(120), IN
newfechaNacimiento DATE, IN newEdad INT, IN newEmail VARCHAR(120), IN newDireccion
VARCHAR(120), IN newTelefono VARCHAR(50),IN cargoDire VARCHAR(120),  IN AnioAlum INT, IN
cursoAlum INT, IN tituProfe VARCHAR(120), IN experProfe INT)
BEGIN
        # INGRESO A UN DIRECTIVO
        IF newIdRol = 1
                THEN
                        BEGIN
                                INSERT INTO USUARIO (dni, idRol, nickname, clave, nombre,
fechaNacimiento, edad, email, direccion, telefono)
                        VALUES (newdni, newidRol, newNickname, newClave, newNombre,
newFechaNacimiento, newEdad, newEmail, newDireccion, newTelefono);
                        CALL spInsertDirectivo(newDni, cargoDire);
                        END;

        # INGRESO A UN PROFESOR
        ELSEIF newIdRol = 2
                        THEN
                                BEGIN
                                        INSERT INTO USUARIO (dni, idRol, nickname, clave,
nombre, fechaNacimiento, edad, email, direccion, telefono)
                                VALUES (newDni, newidRol, newNickname, newClave, newNombre,
newFechaNacimiento, newEdad, newEmail, newDireccion, newTelefono);
                                CALL spInsertProfesor(newDni, tituProfe, experProfe);
         END;
                # INGRESO A UN ALUMNO #newIdRol = 3
        ELSE
                BEGIN
                        INSERT INTO USUARIO (dni, idRol, nickname, clave, nombre,
                fechaNacimiento, edad, email, direccion, telefono)
                        VALUES (newDni, newidRol, newNickname, newClave, newNombre,
newFechaNacimiento, newEdad, newEmail, newDireccion, newTelefono);
                        CALL spInsertAlumno(newDni, AnioAlum, cursoAlum);
                END;
        END IF;
END //
DELIMITER ;
```

```
# sp para eliminar logicamente un usuario
DELIMITER //
DROP PROCEDURE IF EXISTS spDeleteUsuario;
CREATE PROCEDURE spDeleteUsuario(IN deleteDni INT)
BEGIN
		UPDATE USUARIO
		SET activo = FALSE
		WHERE dni = deleteDni;
END //
DELIMITER ;


# sp para Activar logicamente un usuario
DELIMITER //
DROP PROCEDURE IF EXISTS spActivarUsuario;
CREATE PROCEDURE spActivarUsuario(IN deleteDni INT)
BEGIN
		UPDATE USUARIO
		SET activo = TRUE
		WHERE dni = deleteDni;
END //
DELIMITER ;

# SP para insertar un usuario
DELIMITER //
DROP PROCEDURE IF EXISTS spUpdateUsuario;
CREATE PROCEDURE spUpdateUsuario(IN oldDni INT, IN newIdRol INT , IN newNickname
VARCHAR(50), IN newClave VARCHAR(50), IN newNombre VARCHAR(120), IN
newfechaNacimiento DATE, IN newEdad INT, IN newEmail VARCHAR(120), IN newDireccion
VARCHAR(120), IN newTelefono VARCHAR(50), IN cargoDire VARCHAR(120),  IN AnioAlum INT, IN
cursoAlum INT, IN tituProfe VARCHAR(120), IN experProfe INT)
BEGIN
	# Actualizo datos de un DIRECTIVO
	IF newIdRol = 1  THEN  CALL spUpdateDireCargo(oldDni, NULL, cargoDire);

	# Actualizo datos de un PROFESOR
	ELSEIF newIdRol = 2 THEN CALL spUpdateProfeTituExper(oldDni, NULL, tituProfe,
experProfe);
	# Actualizo datos de un ESTUDIANTE
	    ELSE CALL spUpdateAlumAnioCurso(oldDni, NULL, AnioAlum, cursoAlum);
	 END IF;

	BEGIN
		UPDATE USUARIO
		SET nickname = IF(newNickname IS NOT NULL, newNickname, nickname),
		clave = IF(newClave IS NOT NULL, newClave, clave),
		nombre = IF(newNombre IS NOT NULL, newNombre, nombre),
		fechaNacimiento = IF(newfechaNacimiento IS NOT NULL, newfechaNacimiento,
		fechaNacimiento),
		edad = IF(newEdad IS NOT NULL, newEdad, edad),
		email = IF(newEmail IS NOT NULL, newEmail, email),
		direccion = IF(newDireccion IS NOT NULL, newDireccion, direccion),
```

```
                telefono = IF(newTelefono IS NOT NULL, newTelefono, telefono)
                WHERE dni = oldDni;
        END;

END //
DELIMITER ;

# sp para devolver los datos de un usuario
DELIMITER //
#DROP PROCEDURE IF EXISTS spGetUsuario;
CREATE PROCEDURE spGetUsuario(IN dameDni INT)
BEGIN
        SELECT * FROM USUARIO  WHERE dni = dameDni;
END //
DELIMITER ;

# sp para devolver todos los datos de los usuarios
DELIMITER //
#DROP PROCEDURE IF EXISTS spGetAllUsuario;
CREATE PROCEDURE spGetAllUsuario()
BEGIN
        SELECT * FROM USUARIO;
END //
DELIMITER ;

# sp para devolver todos los datos de los usuarios con un tipo de rol
DELIMITER //
#DROP PROCEDURE IF EXISTS spGetAllUsuarioRol;
CREATE PROCEDURE spGetAllUsuarioRol(IN rol INT)
BEGIN
        SELECT * FROM USUARIO WHERE idRol = rol;
END //
DELIMITER ;

/*************************************************************/
/************   SP CURSO MATERIA PROFESOR   *****************/
/*************************************************************/
# sp insertar un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spInsertCursoMateriaProfesor;
CREATE PROCEDURE spInsertCursoMateriaProfesor(IN curso INT, IN materia INT, IN profesor INT)
BEGIN
        INSERT INTO cursoMateriaProfesor (idCurso, idMateria, dniProfesor)
         VALUES  (curso, materia, profesor);
END //
DELIMITER ;
```

```
# sp modificar un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spUpDateCursoMateriaProfesor;
CREATE PROCEDURE spUpDateCursoMateriaProfesor(IN idRegistro INT, IN curso INT, IN materia
INT, IN profesor INT)
BEGIN
	UPDATE cursoMateriaProfesor
	SET	idCurso = IF(curso is not null, curso, idCurso),
		idMateria = IF(materia is not null, materia, idMateria),
		 idProfesor = IF(profesor is not null, profesor, idProfesor)
	where id = idRegistro ;
END //
DELIMITER ;

# sp eliminar un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spEliminarCursoMateriaProfesor;
CREATE PROCEDURE spEliminarCursoMateriaProfesor(IN idRegistro INT)
BEGIN
	DELETE FROM cursoMateriaProfesor where id= idRegistro;
END //
DELIMITER ;

# sp devuelve datos de un registro
DELIMITER //
#DROP PROCEDURE IF EXISTS spGetCursoMateriaProfesor;
CREATE PROCEDURE spGetCursoMateriaProfesor(IN idRegistro INT)
BEGIN
	SELECT * FROM cursoMateriaProfesor where id= idRegistro;
END //
DELIMITER ;

# sp devuelve todos los datos de la table
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAllCursoMateriaProfesor;
CREATE PROCEDURE spGetAllCursoMateriaProfesor()
BEGIN
	SELECT * FROM cursoMateriaProfesor;
END //
DELIMITER ;
```

```
/************************************************************/
/**************  SP PROFESOR EVALUACION  ********************/
/************************************************************/
# sp insertar PROFESOR EVALUACION
DELIMITER //
DROP PROCEDURE IF EXISTS spInsertProfesorEvaluacion;
CREATE PROCEDURE spInsertProfesorEvaluacion(IN evaluacion INT, IN tipo INT, IN curso INT, IN
materia INT, IN profesor INT,IN urlEvaluacionCarga2 VARCHAR(250), IN urlEvaluacionDescarga2
VARCHAR(250), IN nombreEvaluacion2 VARCHAR(120))
BEGIN
        INSERT INTO ProfesorEvaluacion (idEvaluacion, idTipo, idCurso, idMateria, dniProfesor,
urlEvaluacionCarga, urlEvaluacionDescarga, nombreEvaluacion)
        VALUES  (evaluacion, tipo, curso, materia, profesor, urlEvaluacionCarga2,
        urlEvaluacionDescarga2, nombreEvaluacion2);
END //
DELIMITER  ;

# sp modificar un registro ProfesorEvaluacion
DELIMITER //
DROP PROCEDURE IF EXISTS spUpDateProfesorEvaluacion;
CREATE PROCEDURE spUpDateProfesorEvaluacion(IN upDateEvaluacion INT, IN tipo INT, IN curso
INT, IN materia INT, IN profesor INT,IN urlEvaluacionCarga2 VARCHAR(250), IN
urlEvaluacionDescarga2 VARCHAR(250), IN nombreEvaluacion2 VARCHAR(120))
BEGIN
        UPDATE ProfesorEvaluacion
        SET     idTipo = IF(tipo is not null, tipo, idTipo),
                idCurso = IF(curso is not null, curso, idCurso),
                idMateria = IF(materia is not null, materia, idMateria),
                idProfesor = IF(profesor is not null, profesor, idProfesor),
                urlEvaluacionCarga = IF(urlEvaluacionCarga2 is not null, urlEvaluacionCarga2,
                urlEvaluacionCarga),
                urlEvaluacionDescarga = IF(urlEvaluacionDescarga2 is not null,
                urlEvaluacionDescarga2, urlEvaluacionDescarga),
                nombreEvaluacion = IF(nombreEvaluacion2 is not null, nombreEvaluacion2,
                nombreEvaluacion)
        WHERE idEvaluacion = upDateEvaluacion ;
END //
DELIMITER  ;

# sp eliminar un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spEliminarProfesorEvaluacion;
CREATE PROCEDURE spEliminarProfesorEvaluacion(IN idEvaluacion2 INT)
BEGIN
        DELETE FROM ProfesorEvaluacion WHERE idEvaluacion = idEvaluacion2;
END //
DELIMITER ;
```

```
# sp devuelve datos de un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spGetCursoMateriaProfesor;
CREATE PROCEDURE spGetProfesorEvaluacion(IN idEvaluacion2 INT)
BEGIN
	SELECT * FROM ProfesorEvaluacion where  idEvaluacion = idEvaluacion2;
END //
DELIMITER ;

# sp devuelve todos los datos de ProfesorEvaluacion
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAllProfesorEvaluacion;
CREATE PROCEDURE spGetAllProfesorEvaluacion()
BEGIN
	SELECT * FROM ProfesorEvaluacion;
END //
DELIMITER ;

/************************************************************/
/**********  SP ALUMNO EVALUACION  **********************/
/************************************************************/
# sp insertar PROFESOR EVALUACION
DELIMITER //
DROP PROCEDURE IF EXISTS spInsertAlumnoEvaluacion;
CREATE PROCEDURE spInsertAlumnoEvaluacion(IN dniAlumno2 INT, IN idEvaluacion2 INT, IN
notaEvaluacio2 INT)
BEGIN
	INSERT INTO AlumnoEvaluacion (dniAlumno, idEvaluacion, notaEvaluacio)
	VALUES  (dniAlumno2, idEvaluacion2, notaEvaluacio2);
END //
DELIMITER ;

# sp modificar un registro AlumnoEvaluacion
DELIMITER //
DROP PROCEDURE IF EXISTS spUpDateAlumnoEvaluacion;
CREATE PROCEDURE spUpDateAlumnoEvaluacion(IN idAlumnoEvaluacion2 INT, IN dniAlumno2
INT, IN idEvaluacion2 INT, IN notaEvaluacio2 INT)
BEGIN
	UPDATE AlumnoEvaluacion
	SET	dniAlumno = IF(dniAlumno2 is not null, dniAlumno2, dniAlumno),
		idEvaluacion = IF(idEvaluacion2 is not null, idEvaluacion2, idEvaluacion),
		notaEvaluacio = IF(notaEvaluacio2 is not null, notaEvaluacio2, notaEvaluacio)
	WHERE idAlumnoEvaluacion = idAlumnoEvaluacion2;
END //
DELIMITER ;
```

```
# sp eliminar un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spEliminarAlumnoEvaluacion;
CREATE PROCEDURE spEliminarAlumnoEvaluacion(IN idAlumnoEvaluacion2 INT)
BEGIN
	DELETE FROM AlumnoEvaluacion WHERE idAlumnoEvaluacion = idAlumnoEvaluacion2;
END //
DELIMITER ;

# sp devuelve datos de un registro
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAlumnoEvaluacion;
CREATE PROCEDURE spGetAlumnoEvaluacion(IN idAlumnoEvaluacion2 INT)
BEGIN
	SELECT * FROM AlumnoEvaluacion where  idAlumnoEvaluacion = idAlumnoEvaluacion2;
END //
DELIMITER ;

# sp devuelve todos los datos de AlumnoEvaluacion
DELIMITER //
DROP PROCEDURE IF EXISTS spGetAllAlumnoEvaluacion;
CREATE PROCEDURE spGetAllAlumnoEvaluacion()
BEGIN
	SELECT * FROM AlumnoEvaluacion;
END //
DELIMITER ;
```