UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN

"PROYECTO FINAL"

DESARROLLO DE APLICACIONES EN BASES DE DATOS RELACIONELES

P R E S E N T A N

CUEVAS MENDOZA ARELI DEL ROSARIO

MÉNDEZ JASSO ÁNGEL ALEJANDRO

PÉREZ AMAYA RAFAEL

SEGURA RÍOS BRENDA STEPHANIE

P R O F E S O R

ONOFRE MARTÍNEZ ARMANDO

GRUPO 1591

FECHA: *03 de febrero del 2021*

ESPECIFICAICONES PARA EL USO DEL SISTEMA

- Se debe importar el archivo "BK_Inscripciones.sql" en PostgreSQL en una base de datos llamada "Inscripciones". En caso de no poder realizar la importación, se puede ejecutar el script "SCRIPT_Inscripciones.sql"
- Debe existir el usuario *rdbms* con contraseña *rbms2021*** en PostgreSQL.
- Se le deben otorgar todos los privilegios al usuario *rdbms*

REGLAS DEL NEGOCIO

- No puede haber dos personas con el mismo nombre y apellido.
- Una persona sólo puede ser alumno, profesor o usuario.
- Un alumno tiene un número de cuenta único.
- La búsqueda del alumno se realiza por medio de su número de cuenta.
- No se permite la modificación del número de cuenta de un alumno.
- Un profesor tiene un número de trabajador único.
- La búsqueda del profesor se realiza por medio de su número de trabajador.
- No se permite la modificación del número de trabajador.
- Un grupo tiene una clave de grupo única.
- La búsqueda de un grupo se realiza por medio de la clave de grupo.
- No se permite la modificación de la clave del grupo.
- Para crear o actualizar un grupo debe tener un id de periodo y una clave de asignatura existentes, en caso de no existir, se pueden crear en ese momento.
- La consulta de las inscripciones se realiza por medio de la tabla.
- No se debe realizar la inscripción de un alumno con un mismo grupo.
- Cada que se realiza una inscripción a un grupo, el cupo de dicho grupo disminuye.
- Cada que se elimina una inscripción, se guarda en una tabla de respaldo en la base de datos.
- La fecha, hora y host de inscripción se generan automáticamente con la hora del sistema operativo.
- Al registrar un horario, el id del salón debe existir, en caso contrario, se puede generar en ese momento.

FUNCIONES SQL

CRUD DE ALUMNOS	

BUSCAR	`SELECT *`
	`FROM persona INNER JOIN alumno`
	`ON pers_id_persona = alum_id_persona`
	`WHERE alum_num_cuenta LIKE (alum_num_cuenta);`

INSERTAR	`CREATE OR REPLACE FUNCTION insertar_alumno(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_cuenta VARCHAR, generacion INTEGER) RETURNS void AS $$`
	
	`DECLARE`
	`idPersona INT;`
	`BEGIN`
	`INSERT INTO persona (pers_nombre, pers_apaterno, pers_amaterno)`
	`VALUES (nombre, apPaterno, apMaterno);`
	
	`idPersona := (SELECT pers_id_persona`
	`FROM persona`
	`WHERE pers_nombre LIKE (nombre) AND`
	`pers_apaterno LIKE (apPaterno) AND`
	`pers_amaterno LIKE (apMaterno));`
	
	`INSERT INTO alumno (alum_id_persona, alum_num_cuenta, alum_generacion)`
	`VALUES (idPersona, num_cuenta, generacion);`
	`END;`
	`$$ LANGUAGE plpgsql;`

MODIFICAR	`CREATE OR REPLACE FUNCTION actualizar_alumno(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_cuenta VARCHAR, generacion INTEGER) RETURNS void AS $$`
	
	`DECLARE`
	`idPersona INT;`
	`BEGIN`
	`idPersona := (SELECT alum_id_persona`
	`FROM alumno`
	`WHERE alum_num_cuenta = (num_cuenta));`
	
	`UPDATE persona`
	`SET pers_nombre = (nombre), pers_apaterno = (apPaterno), pers_amaterno = (apMaterno)`
	`WHERE pers_id_persona = (idPersona);`
	
	`UPDATE alumno`
	`SET alum_generacion = (generacion)`
	`WHERE alum_num_cuenta = (num_cuenta);`
	
	`END;`
	`$$ LANGUAGE plpgsql;`

ELIMINAR	`CREATE OR REPLACE FUNCTION borrar_alumno(num_cuenta VARCHAR) RETURNS void AS $$`
	
	`DECLARE`
	`idPersona INT;`
	`BEGIN`
	`idPersona := (SELECT alum_id_persona`
	`FROM alumno`
	`WHERE alum_num_cuenta = (num_cuenta));`
	
	`DELETE FROM alumno WHERE alum_num_cuenta = (num_cuenta);`

	DELETE FROM persona WHERE pers_id_persona = (idPersona); END; $$ LANGUAGE plpgsql;

COMPROBAR EXISTENCIA	CREATE OR REPLACE FUNCTION existe_alumno(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_cuenta VARCHAR) RETURNS INT AS $$
	DECLARE
	idPersona INT;
	contador INT;
	BEGIN
	idPersona := (SELECT pers_id_persona
	FROM persona
	WHERE pers_nombre LIKE (nombre) AND
	pers_apaterno LIKE (apPaterno) AND
	pers_amaterno LIKE (apMaterno));
	contador := (SELECT count(alum_id_alumno)
	FROM alumno
	WHERE alum_num_cuenta = (num_cuenta)
	OR alum_id_persona = (idPersona));
	return contador;
	END;
	$$ LANGUAGE plpgsql;

CRUD DE PROFESORES	
BUSCAR	SELECT * FROM persona INNER JOIN profesor ON pers_id_persona = prof_id_persona WHERE prof_num_trabajador LIKE (prof_num_trabajador);

INSERTAR	CREATE OR REPLACE FUNCTION insertar_profesor(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_trabajador VARCHAR, cedula VARCHAR, grado CHAR) RETURNS void AS $$
	DECLARE
	idPersona INT;
	BEGIN
	INSERT INTO persona (pers_nombre, pers_apaterno, pers_amaterno)
	VALUES (nombre, apPaterno, apMaterno);
	
	idPersona := (SELECT pers_id_persona
	FROM persona
	WHERE pers_nombre LIKE (nombre) AND
	pers_apaterno LIKE (apPaterno) AND
	pers_amaterno LIKE (apMaterno));
	
	INSERT INTO profesor (prof_id_persona, prof_num_trabajador,
	prof_cedula, prof_grado)
	VALUES (idPersona, num_trabajador, cedula, grado);
	END;
	$$ LANGUAGE plpgsql;

MODIFICAR	CREATE OR REPLACE FUNCTION actualizar_profesor(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_trabajador VARCHAR, cedula VARCHAR, grado CHAR) RETURNS void AS $$
	DECLARE
	idPersona INT;
	BEGIN
	idPersona := (SELECT prof_id_persona
	FROM profesor
	WHERE prof_num_trabajador = (num_trabajador));
	
	UPDATE persona
	SET pers_nombre = (nombre), pers_apaterno = (apPaterno),
	pers_amaterno = (apMaterno) WHERE pers_id_persona = (idPersona);
	
	UPDATE profesor
	SET prof_cedula = (cedula), prof_grado = (grado)
	WHERE prof_num_trabajador = (num_trabajador);
	END;
	$$ LANGUAGE plpgsql;

ELIMINAR	CREATE OR REPLACE FUNCTION borrar_profesor(num_trabajador VARCHAR) RETURNS void AS $$ DECLARE idPersona INT; BEGIN idPersona := (SELECT prof_id_persona FROM profesor WHERE prof_num_trabajador = (num_trabajador)); DELETE FROM profesor WHERE prof_num_trabajador = (num_trabajador);

	DELETE FROM persona WHERE pers_id_persona = (idPersona); END; $$ LANGUAGE plpgsql;

COMBROBAR EXISTENCIA	CREATE OR REPLACE FUNCTION existe_profesor(nombre VARCHAR, apPaterno VARCHAR, apMaterno VARCHAR, num_trabajador VARCHAR) RETURNS INT AS $$
	DECLARE
	idPersona INT;
	contador INT;
	BEGIN
	idPersona := (SELECT pers_id_persona
	FROM persona
	WHERE pers_nombre LIKE (nombre) AND
	pers_apaterno LIKE (apPaterno) AND
	pers_amaterno LIKE (apMaterno));
	contador := (SELECT count(prof_id_profesor)
	FROM profesor
	WHERE prof_num_trabajador = (num_trabajador) OR
	prof_id_persona = (idPersona));
	return contador;
	END;
	$$ LANGUAGE plpgsql;

CRUD DE GRUPOS	
BUSCAR	SELECT g.grup_id_asignatura, a.asig_nombre, a.asig_plan_estudio, g.grup_id_periodo, pe.peri_fec_inicio, pe.peri_fec_fin, pe.peri_estado, g.grup_id_grupo, g.grup_clave, g.grupo_cupo, p.prof_num_ FROM grupo g INNER JOIN profesor p ON g.grup_id_profesor = p.prof_id_profesor INNER JOIN periodo pe ON g.grup_id_periodo = pe.peri_id_periodo INNER JOIN asignatura a ON g.grup_id_asignatura = a.asig_id_asignatura WHERE g.grup_clave = (caveGrupo);

INSERTAR	CREATE OR REPLACE FUNCTION insertar_grupo(claveProfesor VARCHAR, periodo VARCHAR, claveAsignatura VARCHAR, claveGrupo VARCHAR, cupo INT) RETURNS void AS $$
	DECLARE
	idProfesor INT;
	BEGIN
	idProfesor := (SELECT prof_id_profesor
	FROM profesor
	WHERE prof_num_trabajador = (claveProfesor));
	
	INSERT INTO grupo (grup_id_profesor, grup_id_periodo,
	grup_id_asignatura, grup_clave, grupo_cupo)
	VALUES (idProfesor, periodo, claveAsignatura, claveGrupo, cupo);
	END;
	$$ LANGUAGE plpgsql;

MODIFICAR	CREATE OR REPLACE FUNCTION actualizar_grupo(claveProfesor VARCHAR, periodo VARCHAR, claveAsignatura VARCHAR, claveGrupo VARCHAR, cupo INT) RETURNS void AS $$ DECLARE idProfesor INT; BEGIN idProfesor := (SELECT prof_id_profesor

	FROM profesor WHERE prof_num_trabajador = (claveProfesor)); UPDATE grupo SET grup_id_profesor = (idProfesor), grup_id_periodo = (periodo), grup_id_asignatura = (claveAsignatura), grup_clave = (claveGrupo), grupo_cupo = (cupo) WHERE grup_clave = (claveGrupo); END; $$ LANGUAGE plpgsql;
ELIMINAR	DELETE FROM grupo WHERE grup_clave = (claveGrupo)

CRUD DE INSCRIPCIONES	
BUSCAR	SELECT * FROM inscripcion i INNER JOIN grupo g ON i.insc_id_grupo = g.grup_id_grupo INNER JOIN alumno a ON i.insc_id_alumno = a.alum_id_alumno WHERE a.alum_num_cuenta = (numCuenta) AND g.grup_clave = (grupClave);

INSERTAR	CREATE OR REPLACE FUNCTION insertar_inscripcion(cuentaAlumno VARCHAR, claveGrupo VARCHAR) RETURNS void AS $$
	DECLARE
	idGrupo INT;
	idAlumno INT;
	fecha DATE;
	hora TIME;
	ip VARCHAR;
	BEGIN
	fecha := (SELECT current_date);
	hora := (SELECT current_time);
	IP:= (SELECT inet_client_addr());
	idGrupo := (SELECT grup_id_grupo
	FROM grupo
	WHERE grup_clave = (claveGrupo));
	idAlumno := (SELECT alum_id_alumno
	FROM alumno
	WHERE alum_num_cuenta = (cuentaAlumno));
	INSERT INTO inscripcion
	VALUES (idAlumno, idGrupo, hora, fecha, ip);
	END;
	$$ LANGUAGE plpgsql;

MODIFICAR	CREATE OR REPLACE FUNCTION actualizar_inscripcion(cuentaAlumno VARCHAR, claveGrupo VARCHAR) RETURNS void AS $$
	DECLARE
	idGrupo INT;
	idAlumno INT;
	fecha DATE;
	hora TIME;
	ip VARCHAR;
	BEGIN
	fecha := (SELECT current_date);
	hora := (SELECT current_time);
	IP:= (select inet_client_addr());
	idGrupo := (SELECT grup_id_grupo
	FROM grupo
	WHERE grup_clave = (claveGrupo));
	idAlumno := (SELECT alum_id_alumno
	FROM alumno
	WHERE alum_num_cuenta = (cuentaAlumno));
	UPDATE inscripcion
	SET insc_id_grupo = (idGrupo), insc_hora = (hora), insc_fecha = (fecha),
	insc_host = (ip) WHERE insc_id_alumno = (idAlumno);
	END;
	$$ LANGUAGE plpgsql;

ELIMINAR	CREATE OR REPLACE FUNCTION borrar_inscripcion(cuentaAlumno VARCHAR, claveGrupo VARCHAR) RETURNS void AS $$
	DECLARE
	idGrupo INT;
	idAlumno INT;
	BEGIN
	idGrupo := (SELECT grup_id_grupo
	FROM grupo
	WHERE grup_clave = (claveGrupo));
	idAlumno := (SELECT alum_id_alumno
	FROM alumno
	WHERE alum_num_cuenta = (cuentaAlumno));
	DELETE FROM inscripcion
	WHERE insc_id_alumno = (idAlumno) AND insc_id_grupo = (idGrupo);
	END;
	$$ LANGUAGE plpgsql;

CRUD DE HORARIOS	
BUSCAR	SELECT * FROM horario h INNER JOIN grupo g ON h.hora_id_grupo = g.grup_id_grupo INNER JOIN hora ho ON h.hora_id_hora = ho.hora_id_hora INNER JOIN dia d ON h.hora_id_dia = d.dia_id_dia INNER JOIN salon s ON h.hora_id_salon = s.salo_id_salon WHERE h.hora_id_horario = (idHorario);
INSERTAR	CREATE OR REPLACE FUNCTION insertar_horario(idSalon VARCHAR, claveGrupo VARCHAR, idDia INT, idHora INT) RETURNS void AS $$ DECLARE idGrupo INT; BEGIN idGrupo := (SELECT grup_id_grupo FROM grupo WHERE grup_clave = (claveGrupo)); INSERT INTO horario (hora_id_grupo, hora_id_hora, hora_id_dia, hora_id_salon)

	VALUES (idGrupo, idHora, idDia, idSalon); END; $$ LANGUAGE plpgsql;

MODIFICAR	CREATE OR REPLACE FUNCTION actualizar_horario(idHorario int, idSalon VARCHAR, claveGrupo VARCHAR, idDia INT, idHora INT) RETURNS void AS $$
	DECLARE
	idGrupo INT;
	BEGIN
	idGrupo := (SELECT grup_id_grupo
	FROM grupo
	WHERE grup_clave = (claveGrupo));
	UPDATE horario SET hora_id_grupo = (idGrupo), hora_id_hora = (idHora), hora_id_dia = (idDia), hora_id_salon = (idSalon) WHERE hora_id_horario = (idHorario);
	END;
	$$ LANGUAGE plpgsql;

ELIMINAR	DELETE FROM horario WHERE hora_id_horario = (idHorario)

EXTRA	
BUSCAR ASIGNATURA	SELECT * FROM asignatura WHERE asig_id_asignatura = ?
INSERTAR ASIGNATURA	INSERT INTO asignatura VALUES (?,?,?)
BUSCAR PERIODO	SELECT * FROM periodo WHERE peri_id_periodo = ?
INSERTAR PERIODO	INSERT INTO periodo VALUES (?,?,?,?)