UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERIA
ESCUELA DE CIENCIAS Y SISTEMAS
LABORATORIO BASES DE DATOS

MANUAL TECNICO
APP-EDUCATIVA

SERGIO FELIPE ZAPETA 200715274
CHRISTTOPHER JOSE CHITAY COUTINO 201113851
MARVIN DANIEL RODRIGUEZ FELIZ 201709450

DESCRIPCION GENERAL

De acuerdo con los requerimientos de la Unidad Académica USAC, este manual fue diseñado para ser usado como una referencia. Este no es un tutorial paso por paso. Y está enfocado en brindar información necesaria para realizar mantenimiento y exploración de la “APP-EDUCATIVA”.

El manual ofrece la información necesaria para entender como está construido el software para que el desarrollador (React.js, MySQL, Nodejs) que necesiten dar soporte y mantenimiento al software lo realicen de manera apropiada. También detallar los aspectos técnicos e informativos del Software y así explicar la estructura del aplicativo al personal que quiera administrarlo, editarlo y configurarlo.

ASPECTOS TECNICOS

El sistema de APP-EDUCATIVA cuenta con una arquitectura CLIENTE-SERVIDOR y se divide en varios módulos con funcionalidades específicas para su máximo desempeño de su diseño.

ENTORNOS PARA EL DESARROLLO DEL SISTEMA

1.1 React Js

React permite a desarrolladores el crear una gran variedad de aplicaciones web que pueden cambiar información sin necesidad de recargar la página. El propósito general de React es ser rápido, escala le y simple. Esto funciona solo en interfaces de usuario. Esto corresponde a la vista en el témplate MVC. Este pude ser usado con combinación de otras librerías de JavaScript o Frameworks tales como Angular JS en MVC.

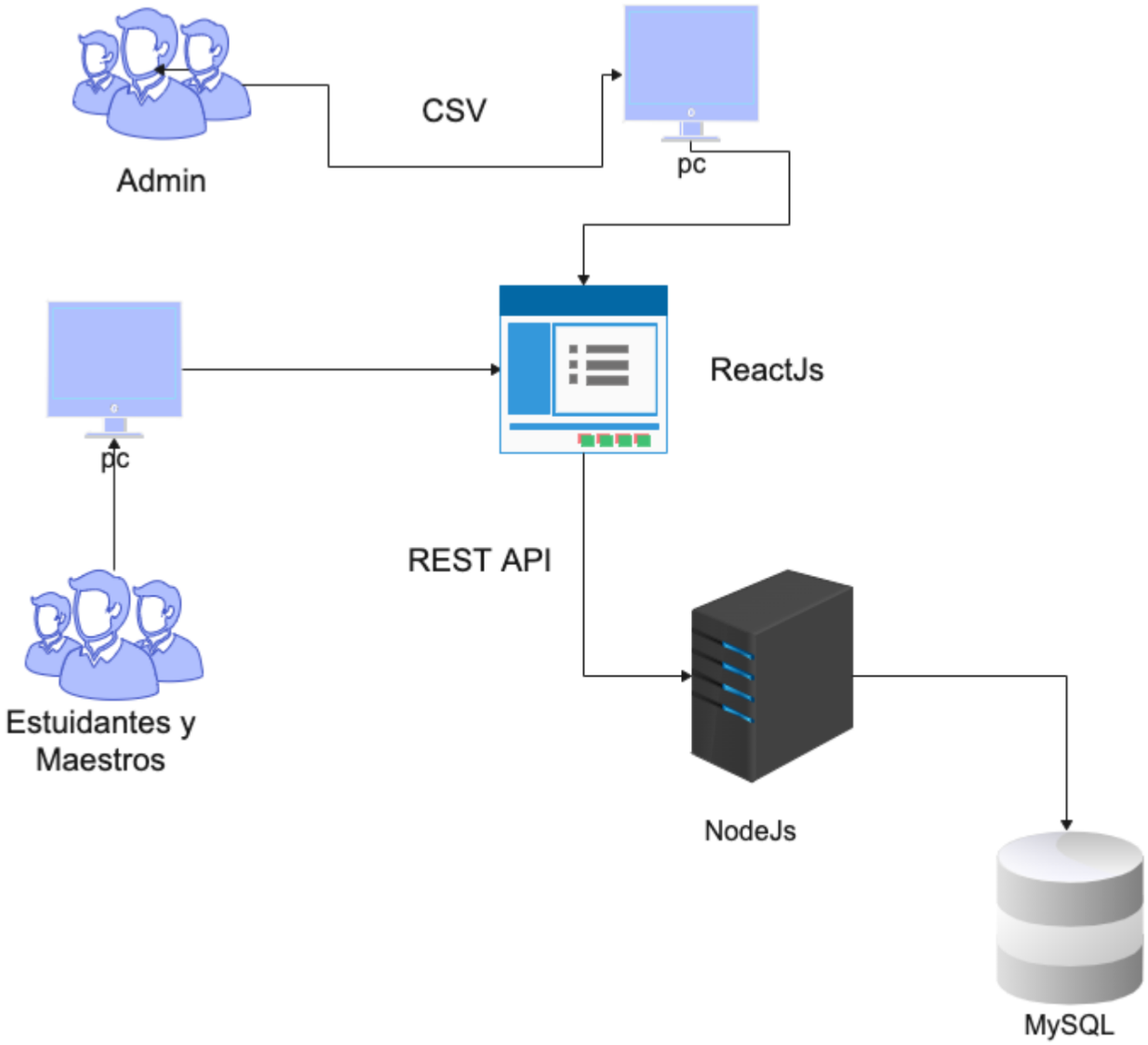
1.2 Nodejs

Node js es primordialmente usado por non-blocking, event-driven servidores, tales como en su simple hilo natural. Este es usado tradicionalmente por web sites y servicios Back-end API, pero fue diseñado con un objetivo de arquitectura real-time, push-based.

1.3 MySQL

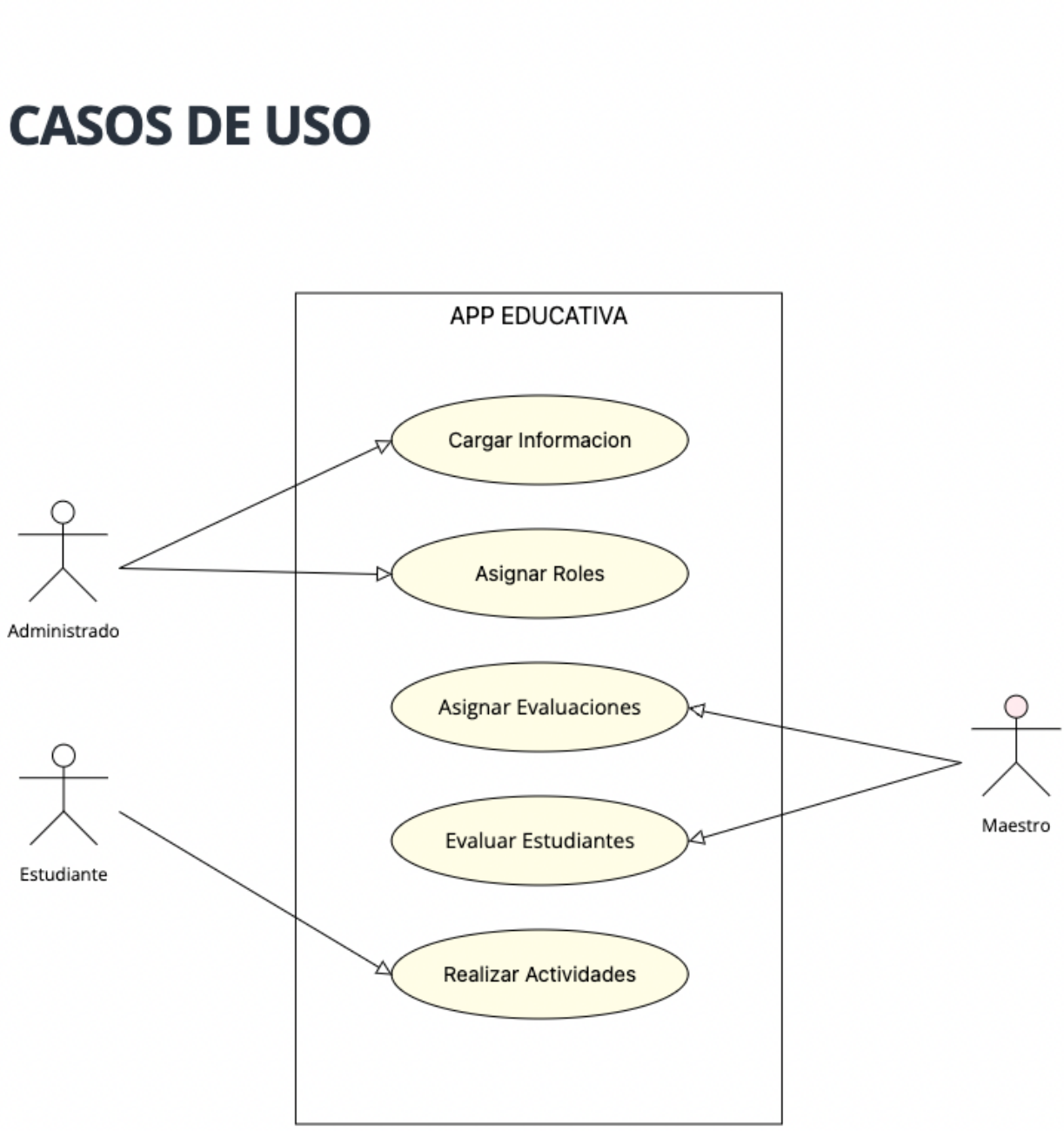
MySQL es un administrador de Bases de Datos. Este es utilizado para agregar, administrar y procesar información en una computadora, y este utiliza un un sistema como MySQL Server. MySQL es ideal para almacenar información para aplicaciones web, adicionalmente se necesita una base de datos relacional que es la que almacenara la información a través de multiples tablas. Como MySQL es una base de datos Relacional, es buena para conectar con aplicaciones que tengan una carga pensada en multi transacciones.

ARQUITECTURA DEL SISTEMA

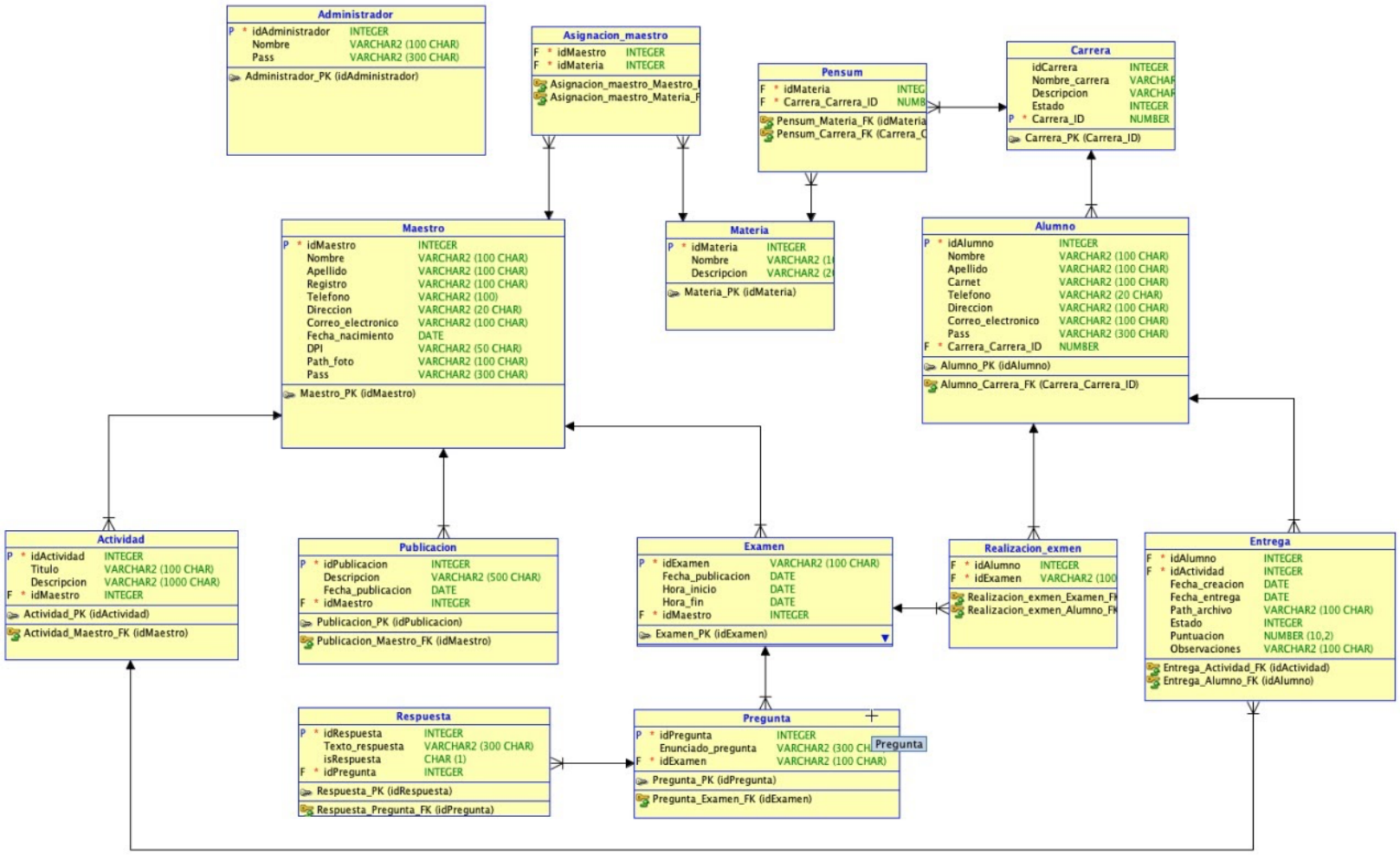


ROLES DEL SISTEMA

En el caso de uso se detalla el rol a desempeñar en relación con la aplicación por parte de las personas relacionadas, en este caso el actor principal es el estudiante, y sus administradores o acompañantes son el administrador y el maestro.



MODELO RELACIONAL



STORE PROCEDURES

Para el desarrollo de la aplicación se han implementado procedimientos almacenados para recuperar, modificar y eliminar información en la base de datos. Para el uso de estos no se necesita escribir una sentencia SQL cada vez que se inserte, actualice o elimine información en la Base de Datos.

A continuación se listan los Procedimientos almacenados utilizados:

Nombre	Tipo	Descripcion	Parametros	Salida
usuario_login	Usuario	se utiliza para realizar el login de cualquier usuario registrado, recibe la iniciall del rol	usuario varchar pass varchar rol char ('A', 'E', 'M')	resp: contiene el id usuario msg_err: contiene el detalle del error
alumno_create	alumno	Se utiliza para la creacion de un alumno dentro de la plataforma	nombre varchar apellido varchar carnet varchar telefono varchar direccion varchar correo varchar pass varchar	resp: contiene el id del alumno msg_err: contiene el detalle del error
alumno_update_delete	alumno	se utiliza para actualizar los campos del alumno excepto el password y el carnet	tipo_operacion int (1 = update, 2= delete) nombre varchar apellido varchar carnet varchar telefono varchar direccion varchar correo varchar estado int	resp: mensaje de exito, msg_err: contiene el detalle del error
alumno_TODO	alumno	obtiene todos los registros de la tabla alumno	offset int page int	Tabla con el numero de resultados segun el valor de offset
alumno_get_by_id	alumno	Busca un alumnosegun el carnet o segun su id. Tiene como prioridad el carnet y como segunda opcion el id.	idAlumno int (enviar 0 si no se tiene) carnet varchar (Enviar '' si no se tiene)	Tabla con 1 o 0 registros encontrados
maestro_create	maestro	Se utiliza para la creacion de un maestro dentro de la plataforma	nombre varchar apellido varchar registro varchar telefono varchar direccion varchar correo varchar fecha_nacimiento date (YYYY/mm/dd)	resp: contiene el id del maestro msg_err: contiene el detalle del error

			dpi varchar path_foto varchar pass varchar	
maestro_update_delete	maestro	se utiliza para actualizar los campos del maestro excepto el password y el registro	tipo_operacion int (1 = update, 2= delete) nombre varchar apellido varchar registro varchar telefono varchar direccion varchar correo varchar fecha_nac date dpi varchar path_foto varchar estado int	resp: mensaje de exito msg_err: contiene el detalle del error
maestro_TODO	maestro	obtiene todos los registros de la tabla maestro	offset int page int	Tabla con el numero de resultados segun el valor de offset
maestro_get_by_id	maestro	Busca un maestro el registro o segun su id. Tiene como prioridad el registro y como segunda opcion el id.	idMaestro int (enviar 0 si no se tiene) registro varchar (Enviar '' si no se tiene)	Tabla con 1 o 0 registros encontrados
carrera_create	carrera	Se utiliza para la creacion de una carrera	nombre_carrera varchar descripcion varchar	resp: contiene el id de la carrera msg_err: contiene el detalle del error
carrera_update_delete	carrera	se utiliza para actualizar los campos de la tabla carrera	tipo_operacion int (1 = update, 2= delete) idCarrera int nombre_carrera varchar descripcion varchar estado int	resp: mensaje de exito msg_err: contiene el detalle del error
carrera_TODO	carrera	obtiene todos los registros de la tabla carrera		taba con todas las carreras existentes
carrera_get_by_id	carrera	Busca una carrera por id o por nombre. Tiene como prioridad el id y como segunda opcion el nombre.	idCarrera int (enviar 0 si no se tiene) Nombre_carrera varchar (Enviar '' si no se tiene)	Tabla con 1 o 0 registros encontrados
materia_create	materia	Se utiliza para la creacion de una materia	nombre varchar descripcion varchar	resp: contiene el id de la materia msg_err: contiene el detalle del error
materia_update_delete	materia	se utiliza para actualizar los campos de la tabla materia	tipo_operacion int (1 = update, 2= delete)	resp: mensaje de exito

			idMateria int nombre_materia varchar descripcion varchar estado int	msg_err: contiene el detalle del error
materia_TODO	materia	obtiene todos los registros de la tabla materia		tabla con todas las materias existentes
materia_get_by_id	materia	Busca una materia por id o por nombre. Tiene como prioridad el id y como segunda opcion el nombre.	idCarrera int (enviar 0 si no se tiene) Nombre_materia varchar (Enviar '' si no se tiene)	Tabla con 1 o 0 registros encontrados
pensum_asignar	pensum	asigna el id enviado al id de la carrera enviada	id_materia int id_carrera int	resp: mensaje de exito msg_err: contiene el detalle del error
pensum_TODO	pensum	Devuelve todas las carreras con todas las materias asignadas		tabla con todas las carreras existentes
materias_get_by_carrera_id	pensum	Devueve todas las materias asignadas a una carrera	id_carrera int	tabla con todas las materias asignadas al id de la carrera enviada
materia_asignar_maestro	asignacion_maestro		id_materia int id_maestro int	resp: mensaje de exito msg_err: contiene el detalle del error
Asignacion_maestro_TODO	asignacion_maestro			tabla con todas las materias asignadas a los maestros
materias_get_by_maestro_id	asignacion_maestro		id_maestro int	tabla con todas las materias asignadas al id maestro enviado
alumno_asignar_carrera	Alumno	Asigna un alumno a una carrera	id_carrera int id_alumno int	resp: mensaje de exito msg_err: contiene el detalle del error
materias_get_by_alumno_id	Alumno	Devuelve las materias asignadas a la carrera del alumno	id_alumno int	tabla con todas las materias asignadas a la carrera
actividad_create	actividad	se utiliza para la creacion de una actividad	titulo varchar descripcion varchar	resp: contiene el id de la materia

			id_materia int fecha_publicacion date(YYYY/mm/dd) fecha_entrega date (YYYY/mm/dd) valor decimal(10,2)	msg_err: contiene el detalle del error
actividad_update_delete	actividad	Se utiliza para actualizar los campos de una actividad o eliminarla	tipo_operacion int (1 = update, 2= delete) idActividad int titulo varchar descripcion varchar id_materia int fecha_publicacion date(YYYY/mm/dd) fecha_entrega date (YYYY/mm/dd) valor decimal(10,2) estado int	resp: mensaje de exito msg_err: contiene el detalle del error
actividad_TODO	actividad	Obtiene todas las actividades disponibles		tabla con todas las actividades creadas
actividad_get_by_id	actividad	Obtiene la actividad con el id enviado	id_actividad	Tabla con 1 o 0 registros encontrados
actividad_get_by_materia_id	actividad	Obtiene todas las actividades disponibles asociadas a una materia	id_materia	Tabla con todas las actividades creadas para la materia enviada como parametro
publicacion_create	publicacion	crea publicaciones	descripcion id materia	resp: contiene el id de la publicacion msg contiene el detalle del error
publicacion_get_by_materia_id	publicacion	Obtiene las publicaciones de la materia	idMateria	tabla con la publicaciones
publicacion_update_delete	publicacion	utiliza para manipular publicaciones	tipo_operacion int (1= update, 2 = delete) idPublicacion descripcion estado	resp: mensaje de exito msg_err: contiene el detalle del error
alumnos_get_by_id_materia	alumnos	se utiliza para obtener todos los alumnos asignados a una materia de su respectiva carrera	idMateria	tabla con los alumnos
actividad_by_alumno	Actividad	se utiliza para obtener las actividades de un alumno	alumno_id INT materia_id INT	Tabla con de actividades

		de una materia en especifico		
actividad_by_alumno	Entrega	se utiliza para la entrega de una actividad del alumno	entrega_id INT path_file VARCHAR	
examen_create	Examen	Utilizado para que un maestro cree un examen asociado a una materia	titulo varchar fecha_publicacion date hora_inicio time hora_fin time id_materia int	resp: mensaje de exito msg_err: contiene el detalle del error
examen_update_delete	Examen	Modifica un examen creado	tipo_operacion int (1= update, 2 = delete) idExamen varchar Fecha_publicacion date Hora_inicio time hora_fin time estado in	resp: mensaje de exito msg_err: contiene el detalle del error
examen_TODO	Examen	obtiene todos los examenes		tabla con todos los examenes
examen_get_by_id	Examen	obtiene un examen en especifico	id_str_Examen varchar	Tabla con 1 o 0 registros encontrados
examen_get_by_materia_id	Examen	obtiene los examenes de una materia	idMateria int	Tabla con el listado de examenes creados para la materia
notificacion_create	Notificacion	Crea una notificacion	titulo varchar contenido varchar idAlumno int idMateria int	resp: mensaje de exito msg_err: contiene el detalle del error
notificacion_update_delete	Notificacion	cambia el estado de una notificacion	id_notificacion int estado int	resp: mensaje de exito msg_err: contiene el detalle del error
notificacion_get_by_alumno_id	Notificacion	obtiene todas las notificaciones asignadas a un alumno	id_alumno int	Tabla con las notificaciones asignados al alumno
notificacion_create	Pregunta	Crea una pregunta	enunciado_pregunta varchar id_str_examen varchar	resp: id de la pregunta creada msg_err: contiene el detalle del error
pregunta_update_delete	Pregunta	Modifica una pregunta creada	tipo_operacion int (1= update, 2 = delete) id_pregunta int	resp: mensaje de exito msg_err:

			enunciado_pregunta varchar estado int	contiene el detalle del error
pregunta_get_by_examen_id	Pregunta	Devuelve todas las preguntas de un examen	id_str_examen varchar	Tabla con lista de preguntas asociadas al examen
respuesta_create	Respuesta	crea una respuesta para una pregunta	texto_respuesta varchar es_respuesta boolean id_pregunta int	resp: id de la respuesta creada msg_err: contiene el detalle del error
respuesta_update_delete	Respuesta	Modica una respuesta	tipo_operacion int (1= update, 2 = delete) id_respuesta int texto_respuesta varchar es_respuesta boolean estado int	resp: mensaje de exito msg_err: contiene el detalle del error
respuesta_get_by_pregunta_id	Respuesta	devuelve las respuestas asociadas a una pregunta	id_pregunta int	Tabla con lista de respuestas asociadas a la pregunta

PROCEDIMIENTO ALMACENADO PARA CREAR ACTIVIDADES

```
DELIMITER //
Drop PROCEDURE if EXISTS actividad_create//
CREATE PROCEDURE actividad_create(par_titulo VARCHAR(100), par_descripcion VARCHAR(100),
par_id_materia int, par_fecha_publicacion date, par_fecha_entrega date, par_valor
decimal(10,2))
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
        DECLARE numresp int;

    select COUNT(*) into numresp from Materia where idMateria = par_id_materia;

        if (par_titulo = '' or  par_descripcion = '' or numresp = 0) THEN
                set msg_err = 'Datos incompletos';
        ELSE
                insert into Actividad(Titulo, Descripcion, idMateria, Fecha_creacion,
Fecha_publicacion, Fecha_entrega, Valor, Estado)
         values (par_titulo, par_descripcion, par_id_materia, now(),
par_fecha_publicacion, par_fecha_entrega, par_valor, 1);

        SELECT LAST_INSERT_ID() into resp;

        Insert into Entrega (IdAlumno, idActividad, Fecha_creacion, Estado)
        Select A.IdAlumno, resp, par_fecha_publicacion, 0 FROM Alumno A
        join Pensum P on P.idCarrera = A.idCarrera
        join Materia M on M.idMateria = P.idMateria
        where M.idMateria = par_id_materia;
```

```
        END IF;

        SELECT msg_err, resp;

END//
DELIMITER ;
```

CREAR USUARIO ADMINISTRADOR

Procedimiento almacenado que permite crear usuarios

```
DELIMITER //
Drop PROCEDURE if EXISTS admin_create//
CREATE PROCEDURE admin_create(par_nombre VARCHAR(100), par_pass VARCHAR(100))
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';

        DECLARE numresp int;

        select COUNT(*) into numresp from Administrador where Nombre = par_nombre;

        if (numresp >0) THEN
                set msg_err = 'usuario ya existe';
        ELSE
                insert into Administrador(Nombre, Pass) values (par_nombre, par_pass);
                SELECT LAST_INSERT_ID() into resp;
        END IF;

        SELECT msg_err, resp;

END //
DELIMITER ;
```

ACTUALIZACIÓN Y ELIMINACIÓN DE ALUMNO

El procedimiento permite actualizar o eliminar alumnos dependiendo los parámetros recibidos.

```
-- Alumno actualizacion y eliminacion
DELIMITER //
Drop PROCEDURE if EXISTS alumno_update_delete//
CREATE PROCEDURE alumno_update_delete(par_tipo_operacion int,
par_nombre VARCHAR(100), par_apellido VARCHAR(100),
par_carnet VARCHAR(100),  par_telefono VARCHAR(100),
par_direccion VARCHAR(100), par_correo VARCHAR(100),
par_estado int)
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
    DECLARE numresp int;

    select COUNT(*) into numresp from Alumno where carnet = par_carnet;

    if( numresp = 1) then
```

```
      if (par_tipo_operacion = 1) THEN
            update Alumno
            set Nombre = par_nombre,
            Apellido = par_apellido, Telefono = par_telefono, Direccion = par_direccion,
Correo_electronico = par_correo
            where carnet = par_carnet;

            set resp = 'datos actualizados correctamente';
        elseif (par_tipo_operacion = 2) THEN
            update Alumno
            set estado = par_estado
            where carnet = par_carnet;

            set resp = 'alumno cambio de estado';
        else
            set msg_err = 'operacion no reconocida';
        end if;
    else
      set msg_err = 'alumno no encontrado';
    end if;

        SELECT msg_err, resp;

END//
DELIMITER ;
```

CREACIÓN DE EXAMEN

```
-- examen: Creacion
DELIMITER //
Drop PROCEDURE if EXISTS examen_create//
CREATE PROCEDURE examen_create(par_titulo VARCHAR(100), par_fecha_publicacion date,
par_hora_inicio time, par_hora_fin time, par_id_materia int)
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
        DECLARE numresp int;

    select COUNT(*) into numresp from Examen where idExamen = par_titulo;

        if (par_titulo = '' or numresp = 1) THEN
                set msg_err = 'Ingresa un titulo diferente';
        ELSE
                insert into Examen(idExamen, Fecha_publicacion, Hora_inicio, Hora_fin,
idMateria, Estado, Fecha_creacion)
        values (par_titulo, par_fecha_publicacion, par_hora_inicio, par_hora_fin,
par_id_materia, 1, now());

        Insert into Realizacion_examen (IdAlumno, idExamen, nota, estado)
        Select A.IdAlumno, par_titulo, 0, 0 FROM Alumno A
        join Pensum P on P.idCarrera = A.idCarrera
        join Materia M on M.idMateria = P.idMateria
        where M.idMateria = par_id_materia;

        set resp = 'Examen creado exitosamente';
        END IF;

        SELECT msg_err, resp;
```

```
END//
DELIMITER ;
```

OBTENER LOS EXÁMENES

```
-- examen obtiene todos los registros de la tabla examen
DELIMITER //
Drop PROCEDURE if EXISTS examen_TODO//
CREATE PROCEDURE examen_TODO()
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
    DECLARE numresp int;

    select e.*, m.Nombre as Nombre_materia from Examen e
    join Materia m on m.idMateria = e.idMateria;

END//
DELIMITER ;
```

CREACIÓN DE NOTIFICACIÓN

El procedimiento crea una nueva notificación que permite almacenar para luego poder ser visualizada por los estudiantes.

```
-- notificacion: Creacion
DELIMITER //
Drop PROCEDURE if EXISTS notificacion_create//
CREATE PROCEDURE notificacion_create(par_titulo VARCHAR(100), par_contenido varchar(100),
par_idAlumno int, par_id_materia int)
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
        DECLARE respAlumno int;
    DECLARE respMateria int;

    select COUNT(*) into respAlumno from Alumno where idAlumno = par_idAlumno;
    select COUNT(*) into respMateria from Materia where idMateria = par_id_materia;

        if (respAlumno = 0 and respMateria = 0) THEN
                set msg_err = 'Datos no encontrados';
        ELSE
                insert into Notificacion(Titulo, Contenido, Fecha_hora_publicacion,
idAlumno, idMateria, Estado)
         values (par_titulo, par_contenido,now(), par_idAlumno, par_id_materia,1);
                set resp = 'notificacion creado exitosamente';
        END IF;

        SELECT msg_err, resp;

END//
```

```
DELIMITER ;
```

CREACIÓN DE PREGUNTA

El procedimiento permite crear preguntas para ser relacionada con un examen.

```
-- pregunta: Creacion
DELIMITER //
Drop PROCEDURE if EXISTS pregunta_create//
CREATE PROCEDURE pregunta_create(par_enunciado varchar(100),par_id_examen varchar(100))
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
        DECLARE intresp int;

    select COUNT(*) into intresp from Examen where idExamen = par_id_examen;


        if (intresp = 0) THEN
                set msg_err = 'Examen no encontrado';
        ELSE
                insert into Pregunta(Enunciado_pregunta, idExamen, Estado)
         values (par_enunciado, par_id_examen,1);
                SELECT LAST_INSERT_ID() into resp;
        END IF;

        SELECT msg_err, resp;

END//
DELIMITER ;
```

CREACIÓN DE RESPUESTA

El procedimiento permite crear respuestas que corresponden a una pregunta de un examen.

```
-- respuesta: Creacion
DELIMITER //
Drop PROCEDURE if EXISTS respuesta_create//
CREATE PROCEDURE respuesta_create(par_texto_respuesta varchar(100), par_es_respuesta
boolean, par_id_pregunta int)
BEGIN

        DECLARE resp VARCHAR(100) DEFAULT '';
        DECLARE msg_err VARCHAR(100) DEFAULT '';
        DECLARE intresp int;

    select COUNT(*) into intresp from Pregunta where idPregunta = par_id_pregunta;


        if (intresp = 0) THEN
                set msg_err = 'Pregunta no encontrado';
        ELSE
```

```
            insert into Respuesta(Texto_respuesta, esRespuesta, idPregunta, Estado)
        values (par_texto_respuesta, par_es_respuesta, par_id_pregunta,1);
            SELECT LAST_INSERT_ID() into resp;
        END IF;

        SELECT msg_err, resp;

END//
DELIMITER ;
```

NODEJS API REST

Una REST API también conocida como RESTful API es una aplicación de programación interfaz o API, que permite interactuar con servicios web.

La siguiente API permite obtener todas las actividades de los estudiantes por materia. Esto en conjunto con el procedimiento almacenado “alumnos_get_by_id_materia(materia_id)” crean la funcionalidad del maestro-ver notas de estudiantes.

```
async function getActividades_students(req,res){
  let data = req.body;
  const alumnos = await db.query(
    `CALL alumnos_get_by_id_materia(${data.materia_id})`
  );
  const students = [];
  await asynForEach(alumnos[0],async (alumno) =>{
    //here i dadd a student info
    const actividades = await db.query(
      `CALL actividad_by_alumno( ${alumno.idAlumno} , ${data.materia_id} );`
    );
    let Notas = []
    let Total = 0;
    actividades[0].map(actividad =>{

      let actividadSchema ={
        "idEntrega":actividad.idEntrega,
        "Path_archivo":actividad.Path_archivo,
        "Estado":actividad.Estado,
        "Puntuacion": parseFloat(actividad.Puntuacion).toFixed(2) ,
        "idActividad":actividad.idActividad,
        "Titulo":actividad.Titulo,
        "Descripcion":actividad.Descripcion,
        "Valor": actividad.Valor,
        "Estado":actividad.Estado
      }

      Notas.push(actividadSchema);
      Total += actividad.Puntuacion;
    })

    const result2 = await db.query(
      `CALL examen_get_by_materia_id_alumno_2(${data.materia_id}, ${alumno.idAlumno})`
    );

    result2[0].map(examen =>{
      Notas.push(examen);
```

```
      Total += examen.Puntuacion;
    });

    let studentSchema = {
      "idAlumno": alumno.idAlumno,
      "Nombre": alumno.Nombre,
      "Apellido": alumno.Apellido,
      "Carnet": alumno.Carnet,
      "Telefono": alumno.Telefono,
      "Direccion":alumno.Direccion,
      "Correo_electronico":alumno.Correo_electronico,
      "Pass": alumno.Pass,
      "Estado": alumno.Estado,
      "idCarrera":alumno.idCarrera,
      "Actividades":Notas,
      "Nota": parseFloat(Total).toFixed(2)
    }
    students.push(studentSchema);
  })
  console.log(students)
  res.contentType('aplication/json').status(200);
  res.send(JSON.stringify(students));
}
```