**SP\_OBTENER\_AREAS**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_OBTENER\_AREAS`(  -- Descripción: Obtiene los edificios relacionados con la carga académica  -- LDeras 2015-07-04    OUT pcMensajeError VARCHAR(1000) -- Parámetro para los mensajes de error  )  BEGIN  DECLARE vcTempMensajeError VARCHAR(1000); -- Variable para anteponer los posibles mensajes de error  DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN    ROLLBACK;    SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError);  SET pcMensajeError := vcTempMensajeError;    END;    -- Obtener los edificios  SET vcTempMensajeError := 'Error al obtener las areas';    SELECT  codigo, nombre  FROM  ca\_areas;    END |

**SP\_INSERTAR\_AREAS**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_INSERTAR\_AREAS`(  IN pcnombre VARCHAR(50), -- Almacena el nombre de la solicitud  OUT pcMensajeError VARCHAR(500) -- Mensaje mostrado el sistema  )  SP:BEGIN    DECLARE vcTempMensajeError VARCHAR(500) DEFAULT ''; -- Variable para almacenar posibles errores no controlados de servidor  DECLARE vnContadorArea INT DEFAULT 0; -- Variable para determinar si el nombre de solicitud ya estÃ¡ siendo usado  DECLARE vcMensajeErrorServidor TEXT; -- Variable para almacenar el mensaje de error del servidor    DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN    ROLLBACK;    GET DIAGNOSTICS CONDITION 1 vcMensajeErrorServidor = MESSAGE\_TEXT;  SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError, ' Error servidor: ', vcMensajeErrorServidor);  SET pcMensajeError := vcTempMensajeError;  END;    -- Determinar si el nombre de solicitud ya estÃ¡ siendo usado  SET vcTempMensajeError := 'Error al seleccionar COUNT de nombre de usuario';  SELECT  COUNT(nombre)  INTO  vnContadorArea  FROM  ca\_areas  WHERE  nombre = pcnombre;      -- El nombre de solicitud ya estÃ¡ siendo usado  IF vnContadorArea > 0 then    SET pcMensajeError := 'El nombre de Solicitud ya esta¡ siendo usado, intenta otro';  LEAVE SP;    END IF;    SET vcTempMensajeError := 'Error al crear el registro en la tabla ca\_facultades';  INSERT INTO ca\_areas (nombre)  VALUES (pcnombre);  END |

**SP\_MODIFICAR\_AREAS**

|  |
| --- |
| -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_MODIFICAR\_AREAS`(  IN pccodigo CHAR(7), -- Almacena el codigo de la solicitud que se va a MODIFICAR  IN pcnombre VARCHAR(50), -- Almacena el nombre de la solicitud  OUT pcMensajeError VARCHAR(500) -- Mensaje mostrado el sistema  )  BEGIN  DECLARE vcTempMensajeError VARCHAR(500) DEFAULT ''; -- Variable para posibles errores no controlados  DECLARE vcMensajeErrorServidor TEXT; -- Variable para almacenar el mensaje de error del servidor    DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN  ROLLBACK;  GET DIAGNOSTICS CONDITION 1 vcMensajeErrorServidor = MESSAGE\_TEXT;  SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError, ' Error servidor: ', vcMensajeErrorServidor);  SET pcMensajeError := vcTempMensajeError;  END;    SET vcTempMensajeError := 'Error al MODIFICAR el registro en la tabla ca\_facultads';  UPDATE ca\_areas SET nombre=pcnombre  WHERE  codigo = pccodigo;  END |

**SP\_ELIMINAR\_AREAS**

|  |
| --- |
| -- --------------------------------------------------------------------------------  -- Routine DDL  -- Note: comments before and after the routine body will not be stored by the server  -- --------------------------------------------------------------------------------  DELIMITER $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `SP\_ELIMINAR\_AREAS`(  IN pnCodigoArea INT, -- Código de area (En caso de que acción sea actualizar o eliminar)  OUT pcMensajeError VARCHAR(1000) -- Parámetro para los mensajes de error  )  SP: BEGIN  DECLARE vcTempMensajeError VARCHAR(1000); -- Variable para anteponer los posibles mensajes de error  DECLARE EXIT HANDLER FOR SQLEXCEPTION  BEGIN  ROLLBACK;    SET vcTempMensajeError := CONCAT('Error: ', vcTempMensajeError);  SET pcMensajeError := vcTempMensajeError;  END;  -- Determinar si la area tiene vinculacion con algun proyecto  SET vcTempMensajeError := 'Error determinar si el area tiene vinculación con algún proyecto';    IF EXISTS  (  SELECT cod\_area  FROM ca\_proyectos  WHERE cod\_area = pnCodigoArea  )  THEN  BEGIN    SET pcMensajeError := 'Hay proyectos que estan viculados con esta area, no puede ser borrada.';  LEAVE SP;    END;  END IF;    -- Eliminar el area  SET vcTempMensajeError := 'Error al eliminar el area';    START TRANSACTION;    DELETE FROM ca\_areas  WHERE codigo = pnCodigoArea;    COMMIT;  end |